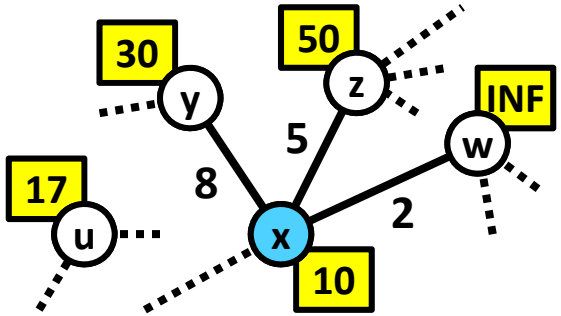
Notes on Minimum Spanning Tree problem

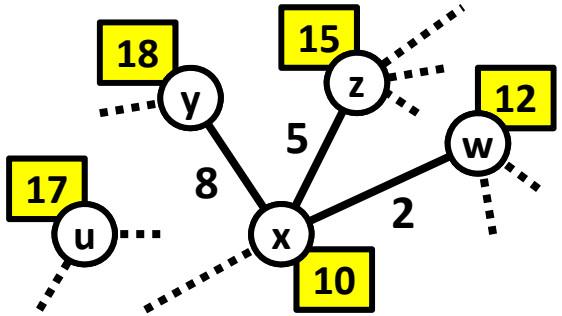
A Priority queue trouble

Usual theoretical and conceptual descriptions of Prim's (Dijkstra's, etc.) algorithm say:
"... store nodes in a priority queue".
Technically, this is nearly impossible to do.
The graph and the nodes are defined separately and stored elsewhere in the memory.
The node has no reference to its position in the queue.
The programmer does not know where is the node in the queue.
So, how to move a node inside the queue according to the algorithm demands?
Standard solution:
Do not move a node, enqueue a "copy of a node", possibly more times.

When a copy of the node with the smallest value (=highest priority) among all its copies appears at the top of the queue it does its job exactly according to the algorithm prescription.
From that moment on, all other copies of the node which are still in the queue become useless and must be ignored. The easiest way to ignore a copy is to check it when it later appears at the top of the queue: If the node is already closed, ignore the copy, pop it and process the next top of the queue. If the node is still open, process it according to the algorithm.



Easy



x	..	...	u	...	y	...	z	...	w	...
10	..	...	17	...	30	...	50	...	INF	...

Difficult

...	..	w	...	z	...	u	...	y	...
...	..	12	...	15	...	17	...	18	...

Easy

q.insert(y);
q.insert(z);
q.insert(w);

..	w	...	z	...	u	...	y	...	y	...	z	...	w	...
..	12	...	15	...	17	...	18	...	30	...	50	...	INF	...

// push the nodes once more to the queue.

The older copies of nodes will get to the top of the queue later than the new copies (which have higher priority) . The older copy gets to the top when the node had been processed and closed earlier. Thus:
If the node at the top of the queue is closed just pop it and do not process it any more.

Example of Prim algorithm implementation using standart library priority queue

```
void MST_Prim(Graph g, int start, final int [] dist, int [] pred ) {
  // allocate structures
  int currnode = start, currdist, neigh;
  int INF = Integer.MAX_VALUE;
  boolean [] closed = new boolean[g.N];

  PriorityQueue <Integer> pq
    = new PriorityQueue<Integer>( g.N,
          new Comparator<Integer>() {
            @Override
            public int compare(Integer n1, Integer n2) {
              if( dist[n1] < dist[n2] ) return -1;
              if( dist[n1] > dist[n2] ) return 1;
              return 0;
          } }  );

  // initialize structures
  pq.add( start );
  for( int i = 0; i < g.N; i++ ) pred[i] = i;
  Arrays.fill( dist, INF );
  Arrays.fill( closed, false );
  dist[start] = 0;
```

Example of Prim algorithm implementation using standart library priority queue

```
  for( int i = 0; i < g.N; i++ ) {
    // take the closest node and skip the closed ones
    while( closed[currnode = pq.poll()] == true );
    // and expand the closest node
    for( int j = 0; j < g.dg[currnode]; j++ ){
      neigh = g.edge[currnode][j];
      if( !closed[neigh] &&
         ( dist[neigh] > g.w[currnode][j]) ) {
           dist[neigh] = g.w[currnode][j];
           pred[neigh] = currnode;
           pq.add(neigh);
      }
     closed[currnode] = true;
    }
  }
}
```

A very small change produces Dijkstra's algorithm:

```
      if( !closed[neigh] &&
         ( dist[neigh] > g.w[currnode][j] + dist[currnode]) ) {
           dist[neigh] = g.w[currnode][j] + dist[currnode];
```

Prim and Dijkstra Algorithms compared

```
void MST_Prim( Graph G, function weight, Node startnode )
  for each u in G.V:  u.dist = INFINITY; u.parent = NIL
  startnode.dist = 0; PriorityQueue Q = G.V

  while !Q.isEmpty()
    u = Extract-Min(Q)
    for each v in G.Adj[u]
      if (v in Q) and v.dist > weight(u,v)
        v.parent = u
        v.dist = weight(u,v)
```

```
void Dijkstra( Graph G, function weight, Node startnode )
  for each u in G.V:  u.dist = INFINITY; u.parent = NIL
  startnode.dist = 0; PriorityQueue Q = G.V

  while !Q.isEmpty()
    u = Extract-Min(Q)
    for each v in G.Adj[u]
      if (v in Q) and v.dist > weight(u,v) + u.dist
        v.parent = u
        v.dist = weight(u,v) + u.dist
```

Disjoint-set data structure alias Union-Find structure

Only 3 operations are needed:

Initialize()

Union(representativeA, representativeB) // merges the two sets represented
// by the given two representatives

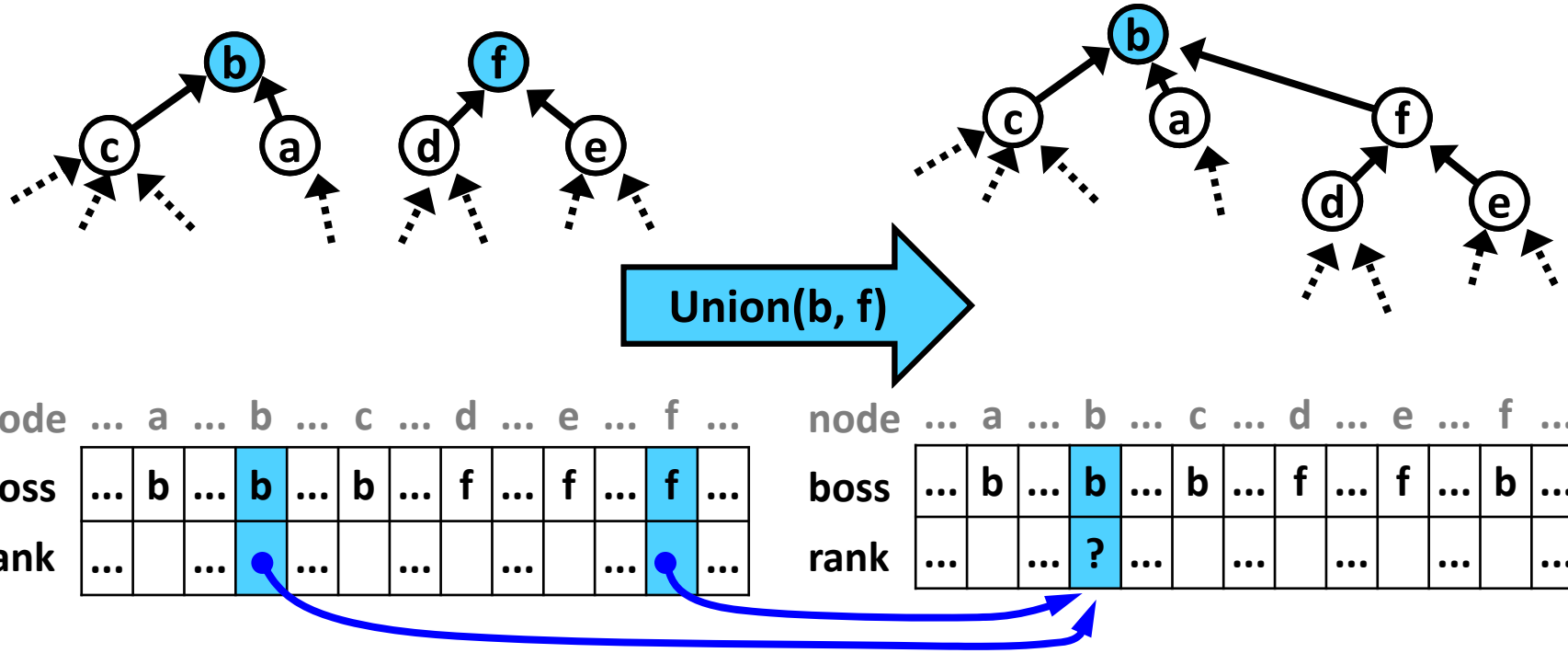
Find(nodeX) // returns a representative of the set to which X belongs

```
int [] boss;
int [] rank;

void UF_init( int n ) {
  boss = new int [n];
  rank = new int [n];
  for( int i = 0; i < n; i++ ) {
    boss[i] = i;   // everybody's their own boss
    rank[i] = 0;   // necessary?
} }
```

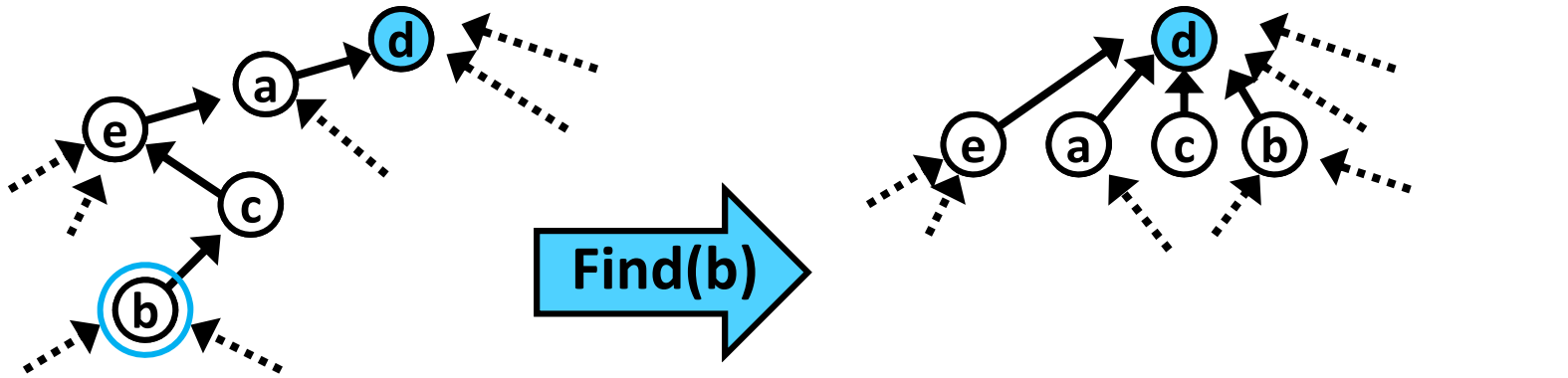
Easy experiment, try it at home:
When the end nodes of the inspected edges are chosen uniformly randomly
then the average depth of a queried node in the Union-Find forest is less than 2.

Union with rank comparison




```
void UF_union( int rootA, int rootB ) {
  if( rank[rootB] > rank[rootA] )
      boss[rootA] = rootB;
  else {
    boss[rootB] = rootA;
    if( rank[rootB] == rank[ rootA ] )   // change rank?
      rank[rootA]++;
} }
```

Find with path compression



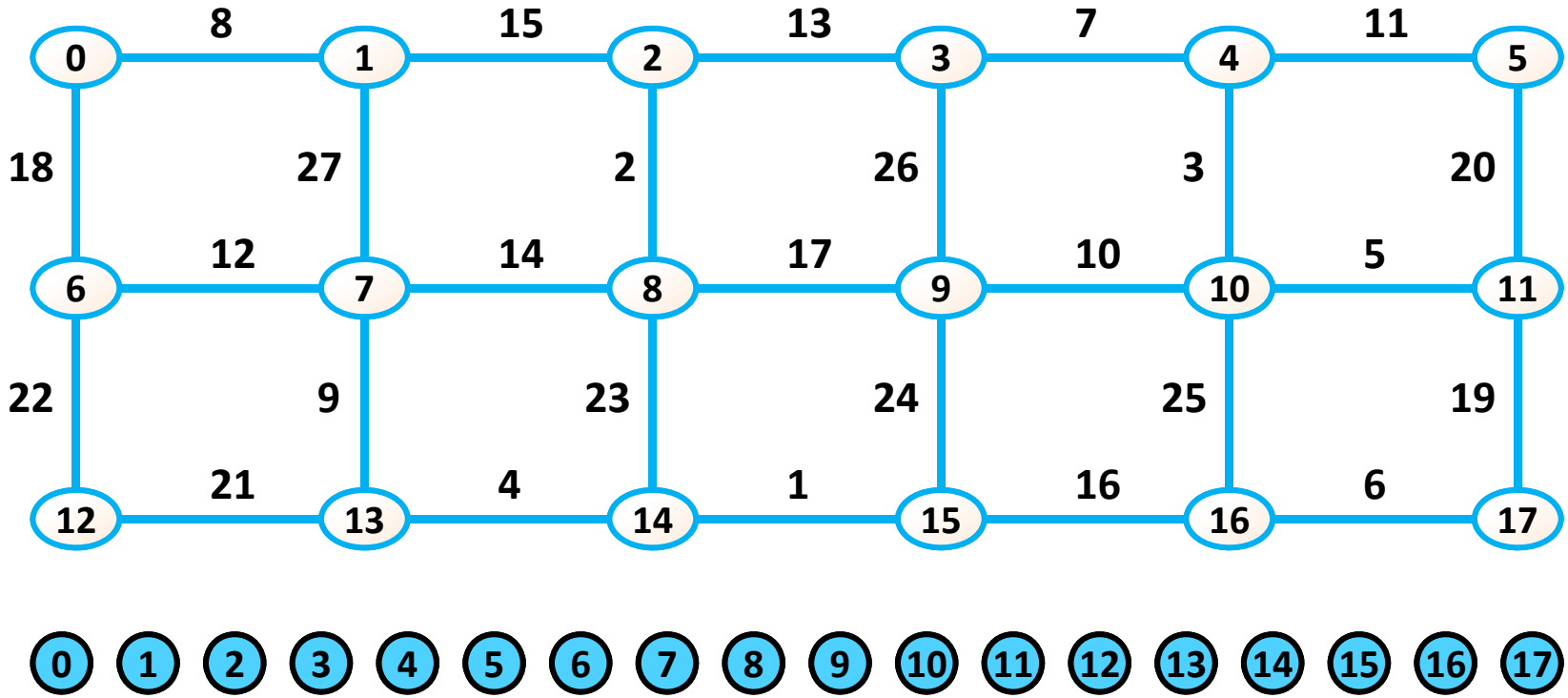
node	…	a	…	b	…	c	…	d	…	e	…
boss	…	d	…	c	…	e	…	d	…	a	…

node	…	a	…	b	…	c	…	d	…	e	…
boss	…	d	…	d	…	d	…	d	…	d	…

```
int UF_find( int a ) {
  int parent = boss[a];
  if( parent != a )
    boss[a] = UF_find( parent ); // path compression
  return boss[a];
}
```

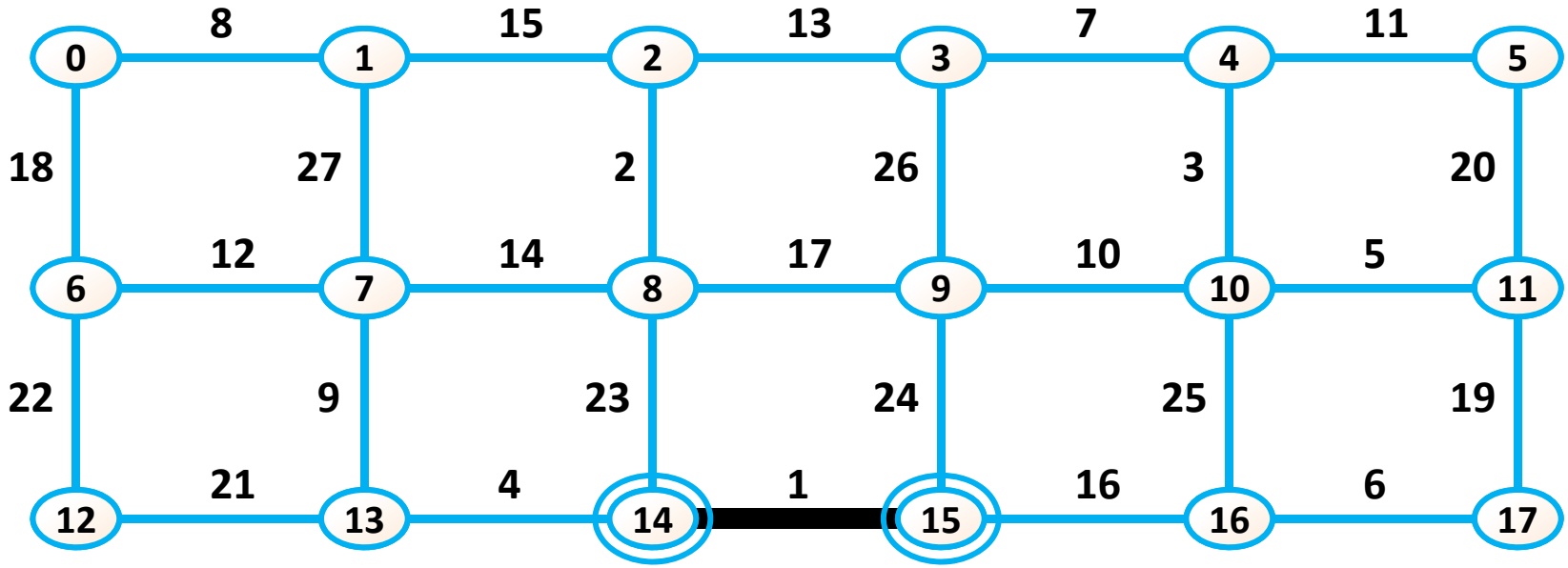
```
int find( int a )  // for C experts
 { return( boss[a] == a ? a : (boss[a] = find(boss[a])) );}
```

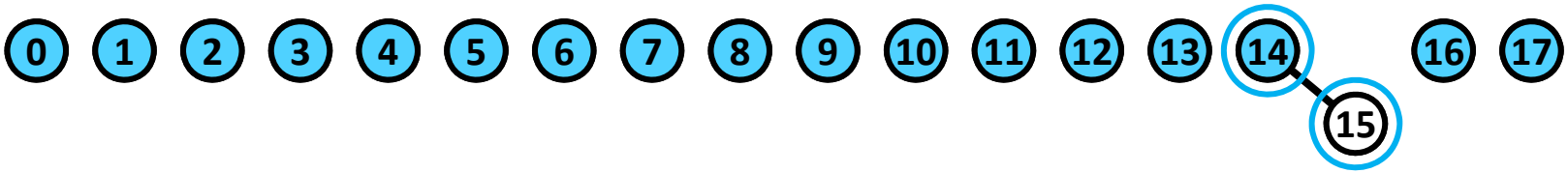
Kruskal algorithm and Union-Find scheme example



node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
rank	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

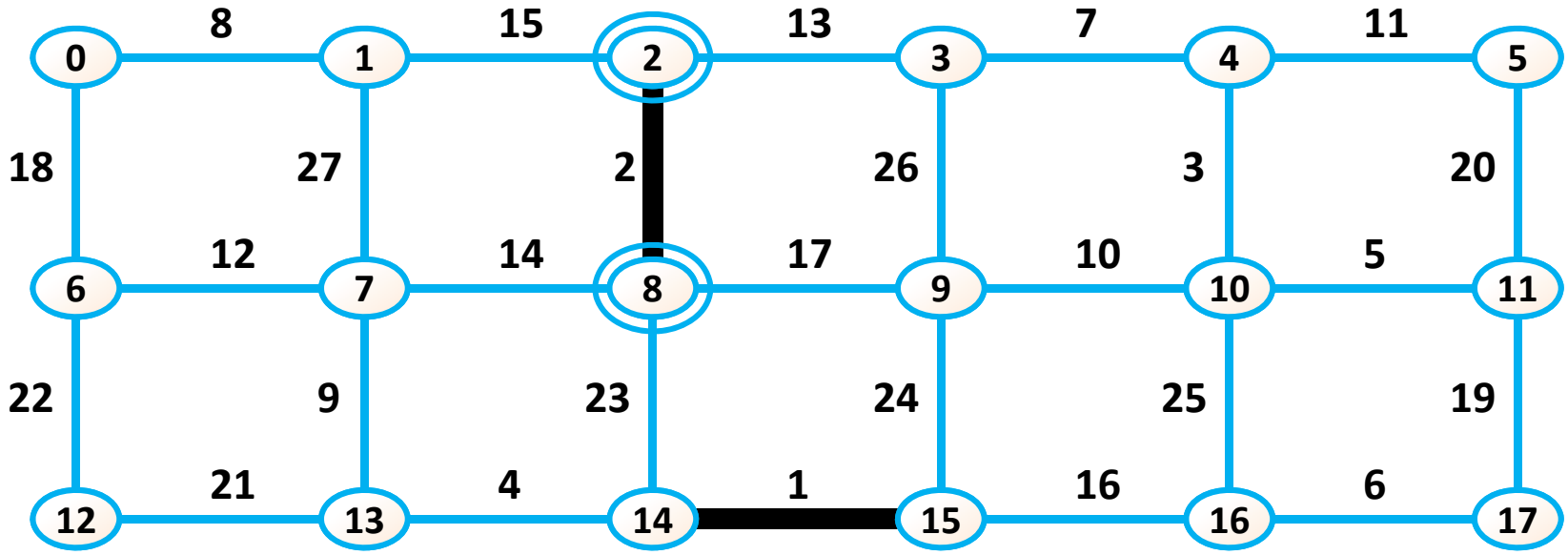
Kruskal algorithm and Union-Find scheme example

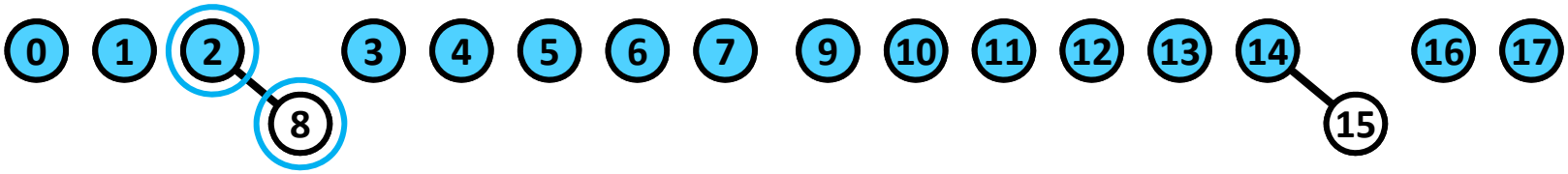




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	14	16	17
rank	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

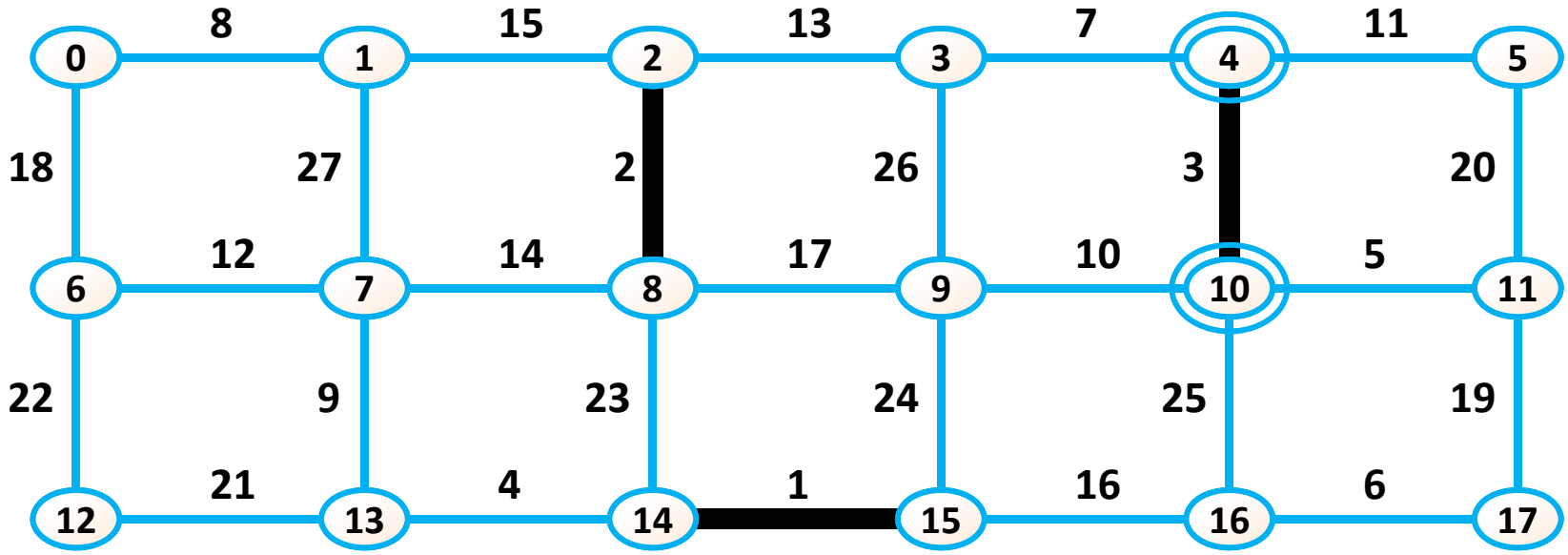
Kruskal algorithm and Union-Find scheme example

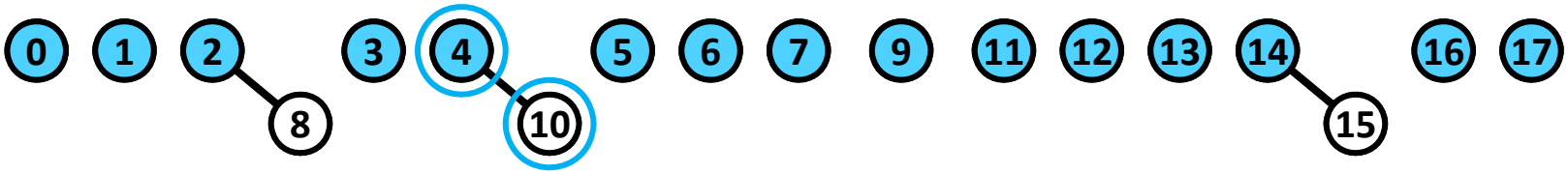




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	10	11	12	13	14	14	16	17
rank	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

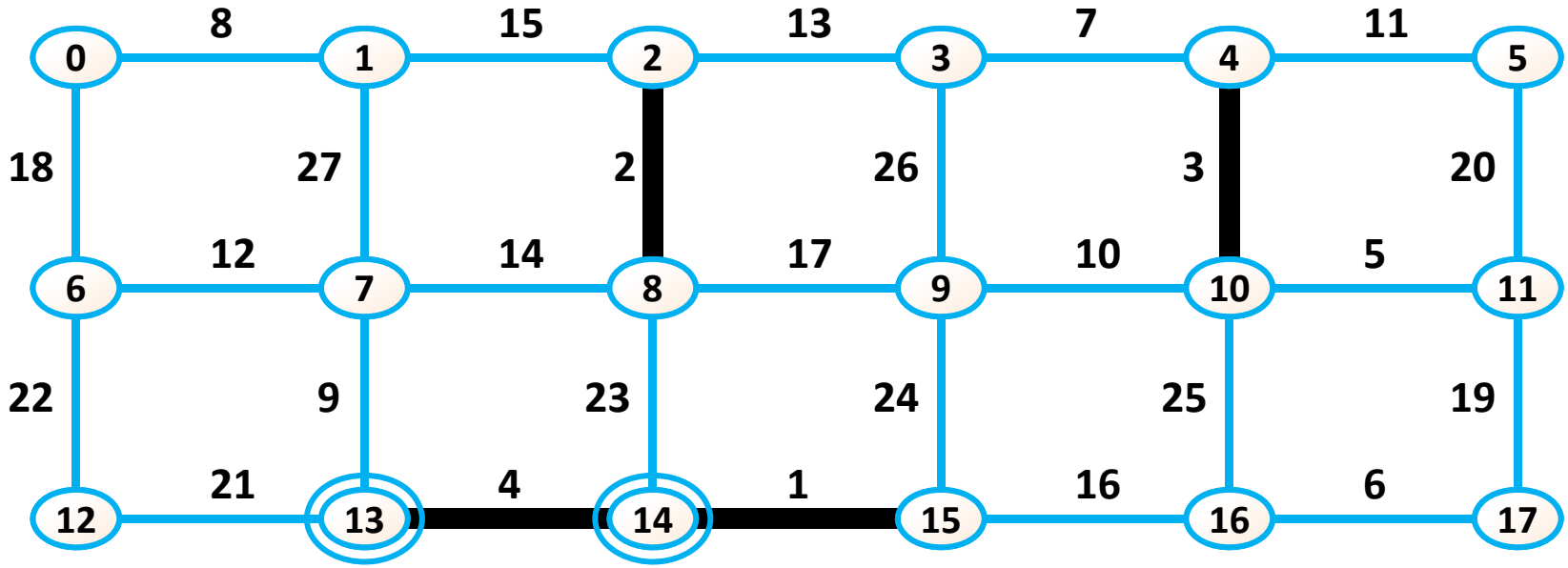
Kruskal algorithm and Union-Find scheme example

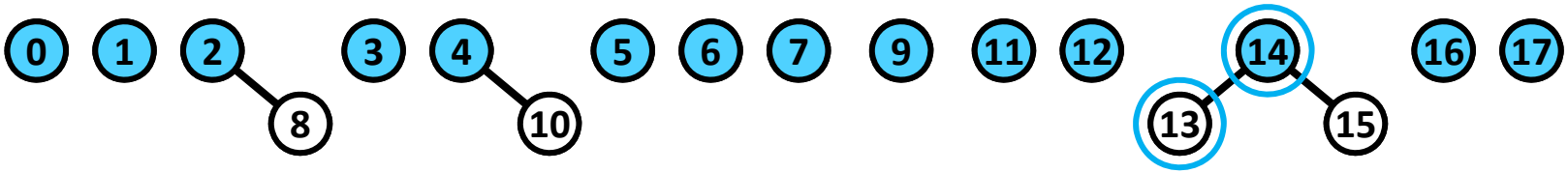




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	4	11	12	13	14	14	16	17
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0

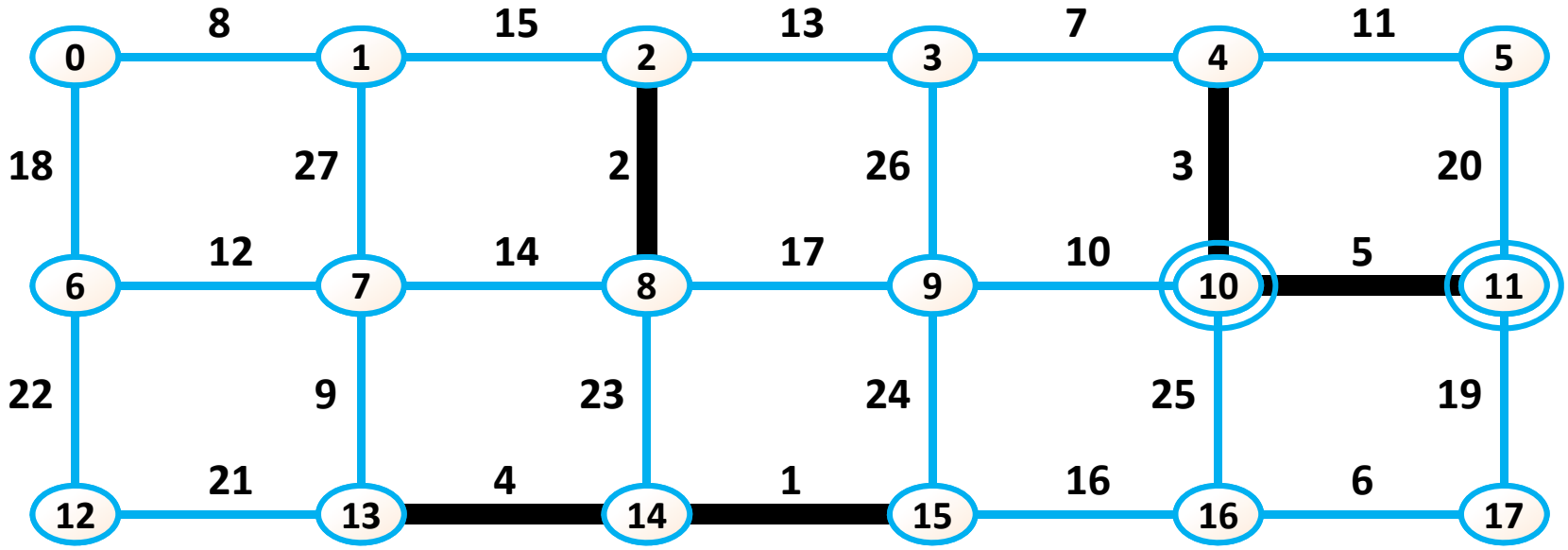
Kruskal algorithm and Union-Find scheme example

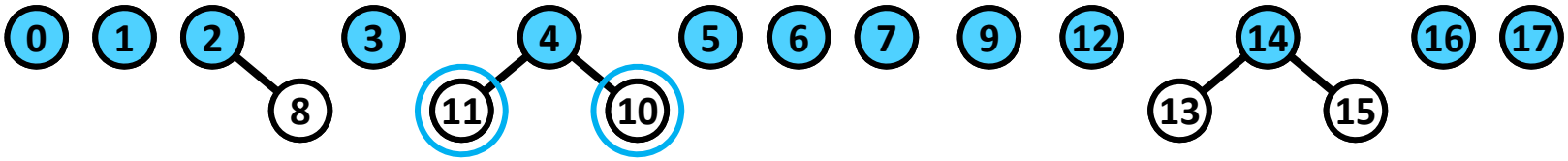




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	4	11	12	14	14	14	16	17
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0

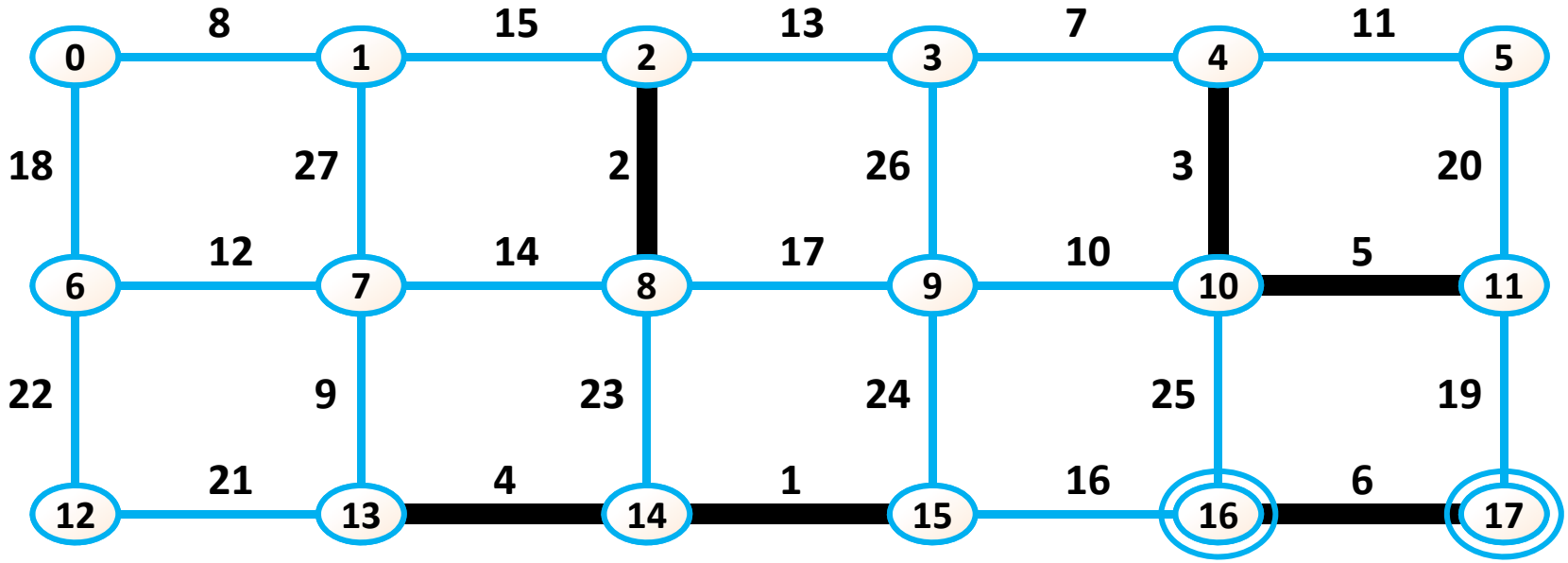
Kruskal algorithm and Union-Find scheme example

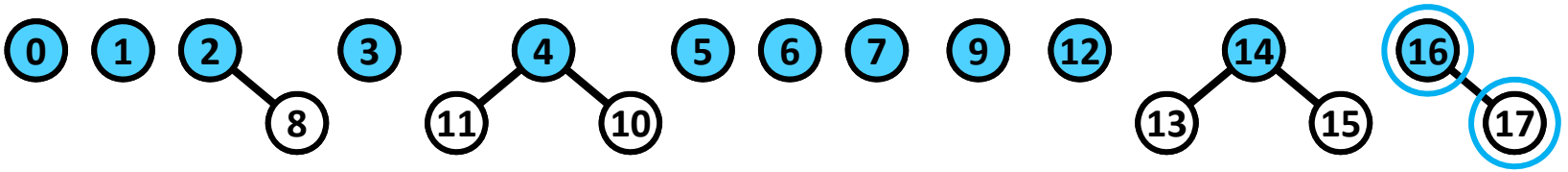




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	4	4	12	14	14	14	16	17
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0

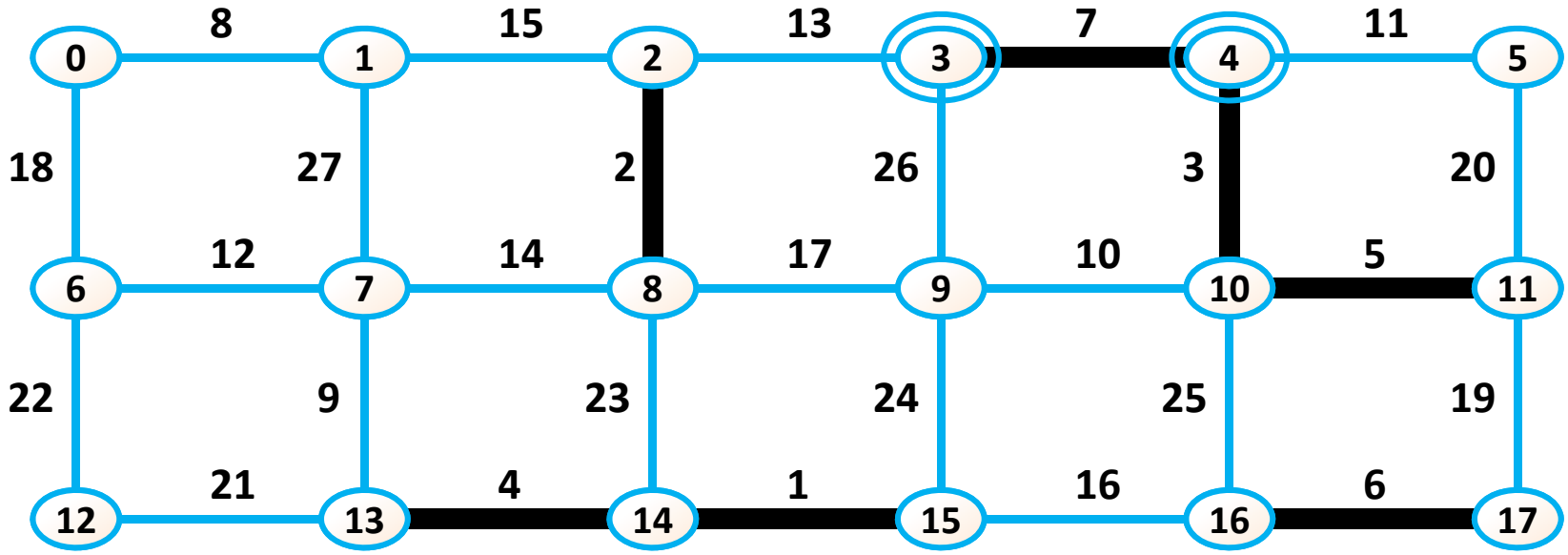
Kruskal algorithm and Union-Find scheme example

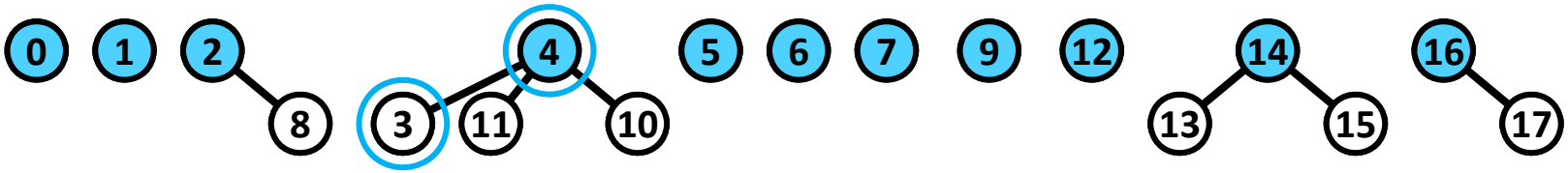




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	4	4	12	14	14	14	16	16
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

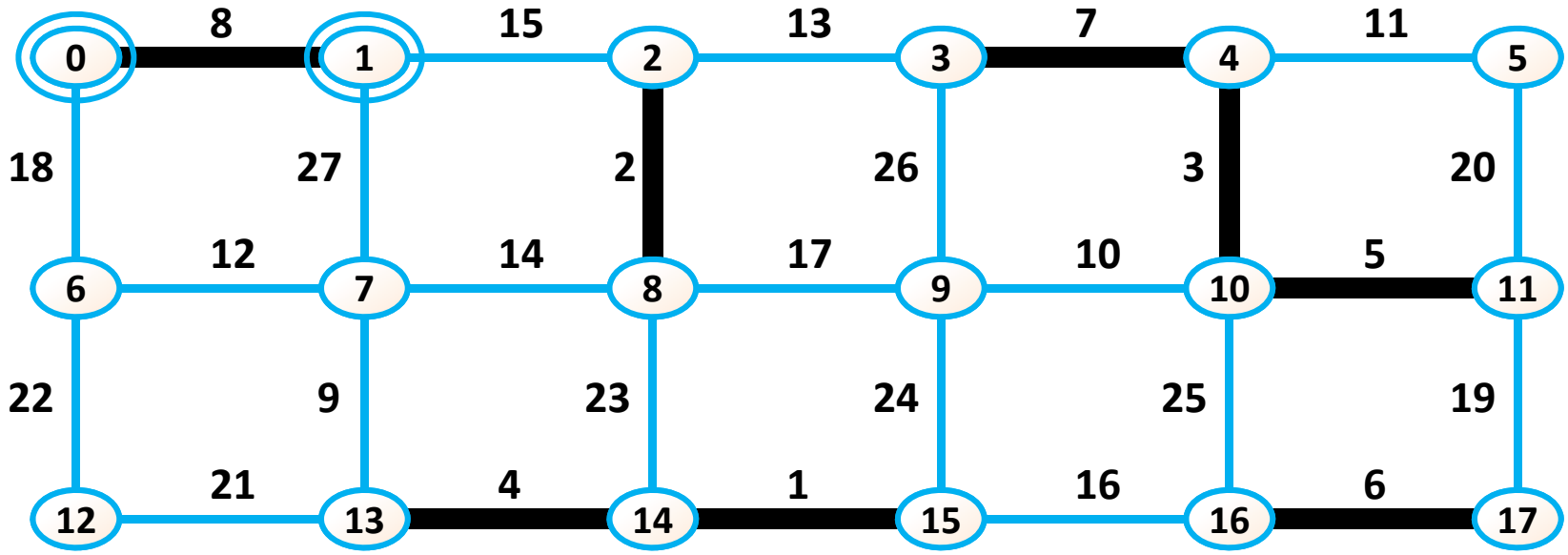
Kruskal algorithm and Union-Find scheme example

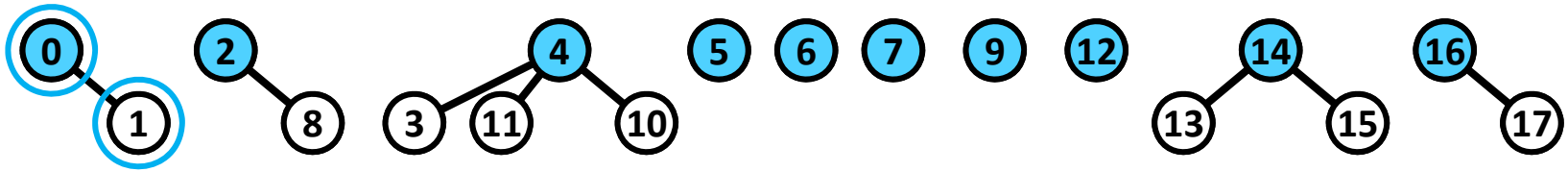




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	4	4	5	6	7	2	9	4	4	12	14	14	14	16	16
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

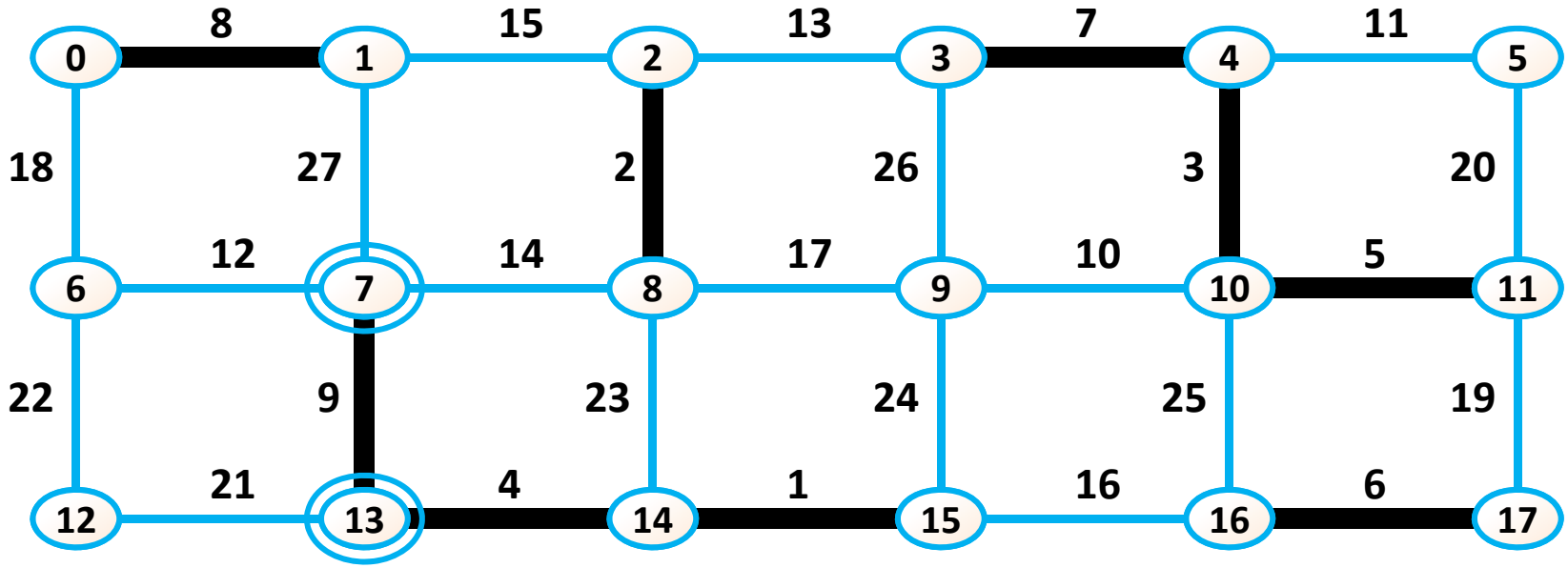
Kruskal algorithm and Union-Find scheme example

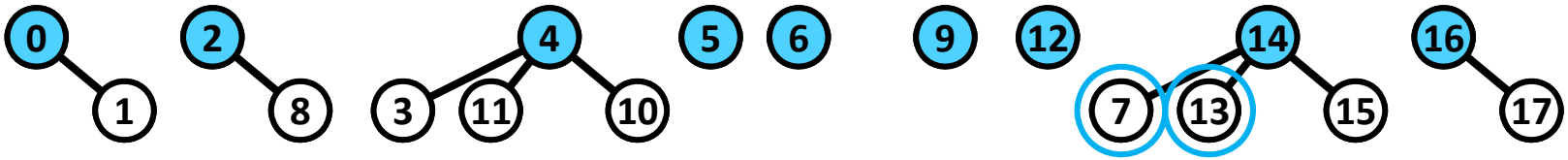




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	5	6	7	2	9	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

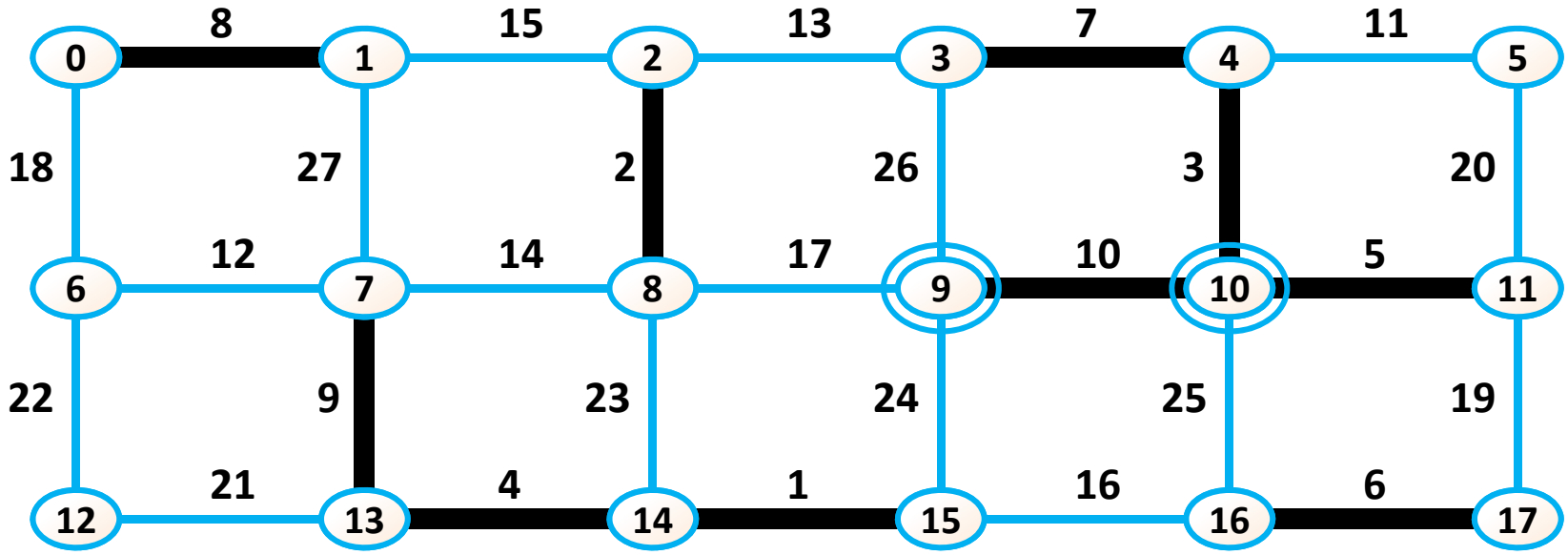
Kruskal algorithm and Union-Find scheme example

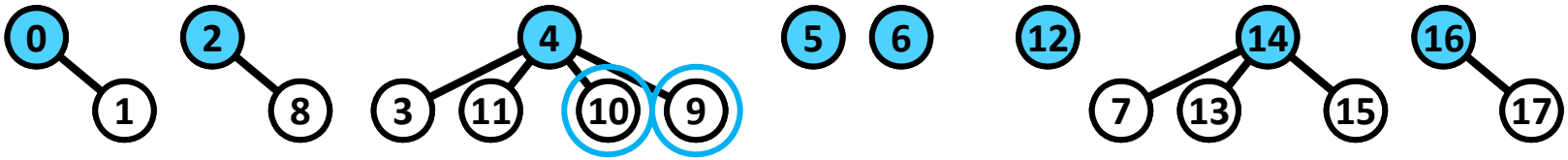




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	5	6	14	2	9	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

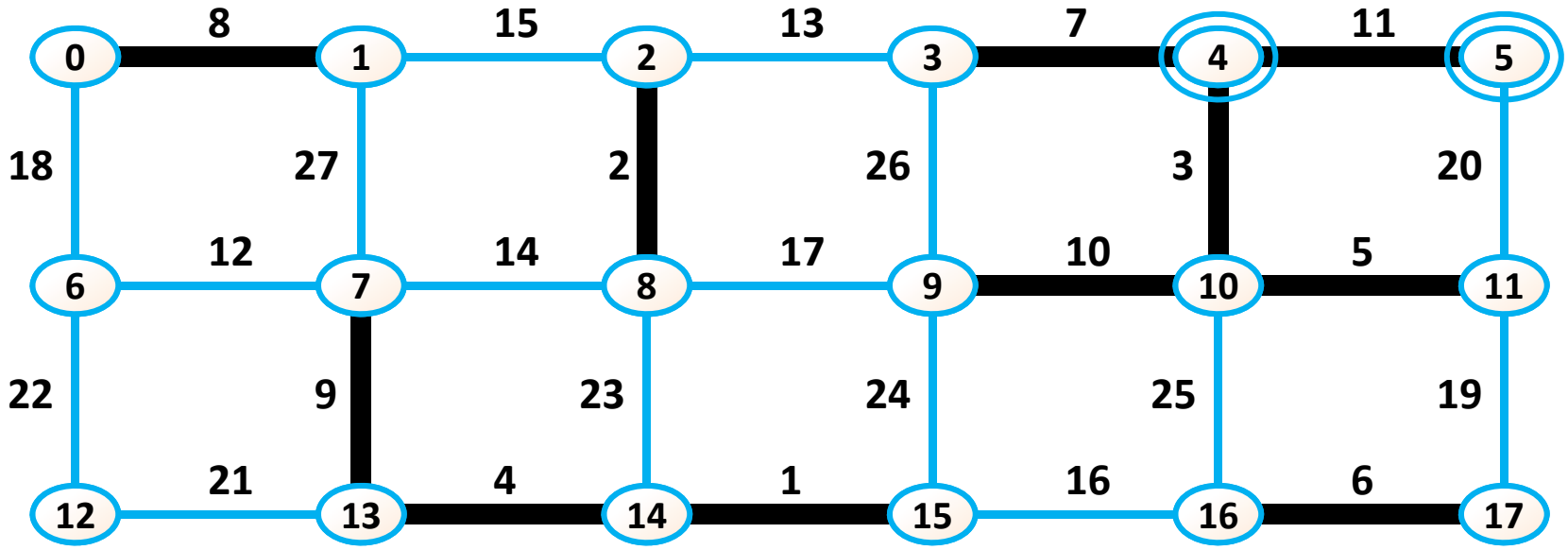
Kruskal algorithm and Union-Find scheme example

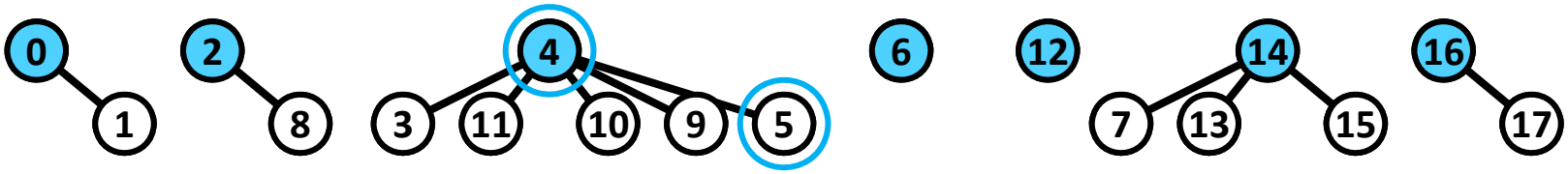




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	5	6	14	2	4	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

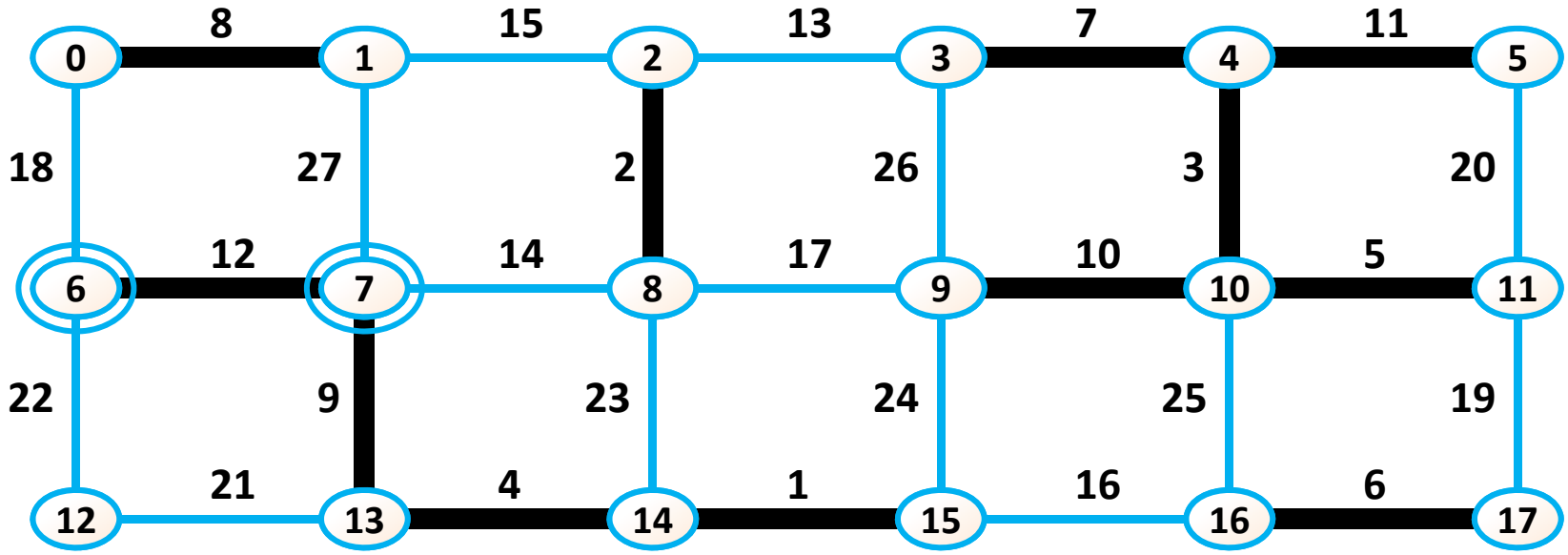
Kruskal algorithm and Union-Find scheme example

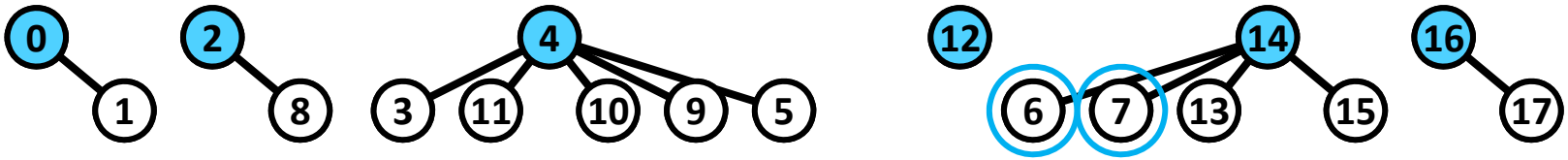




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	4	6	14	2	4	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

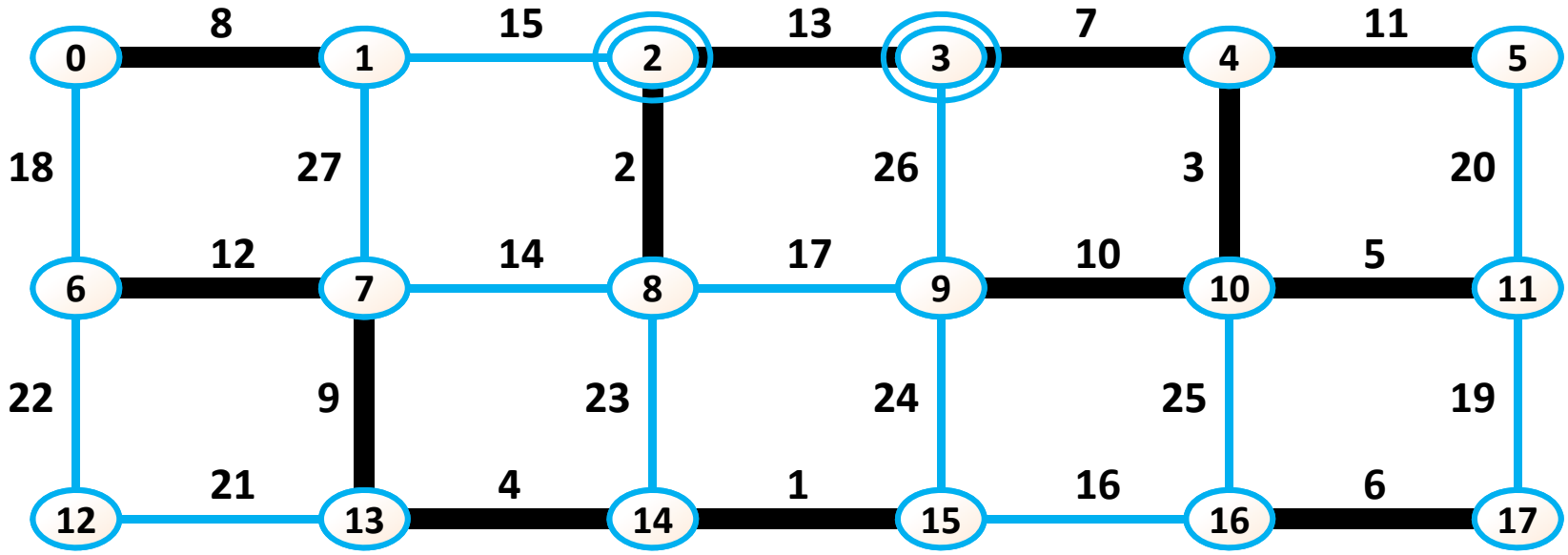
Kruskal algorithm and Union-Find scheme example

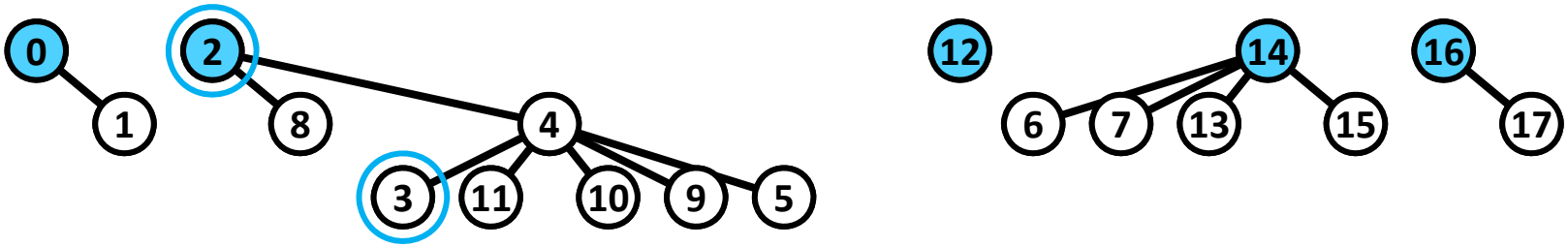




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	4	14	14	2	4	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

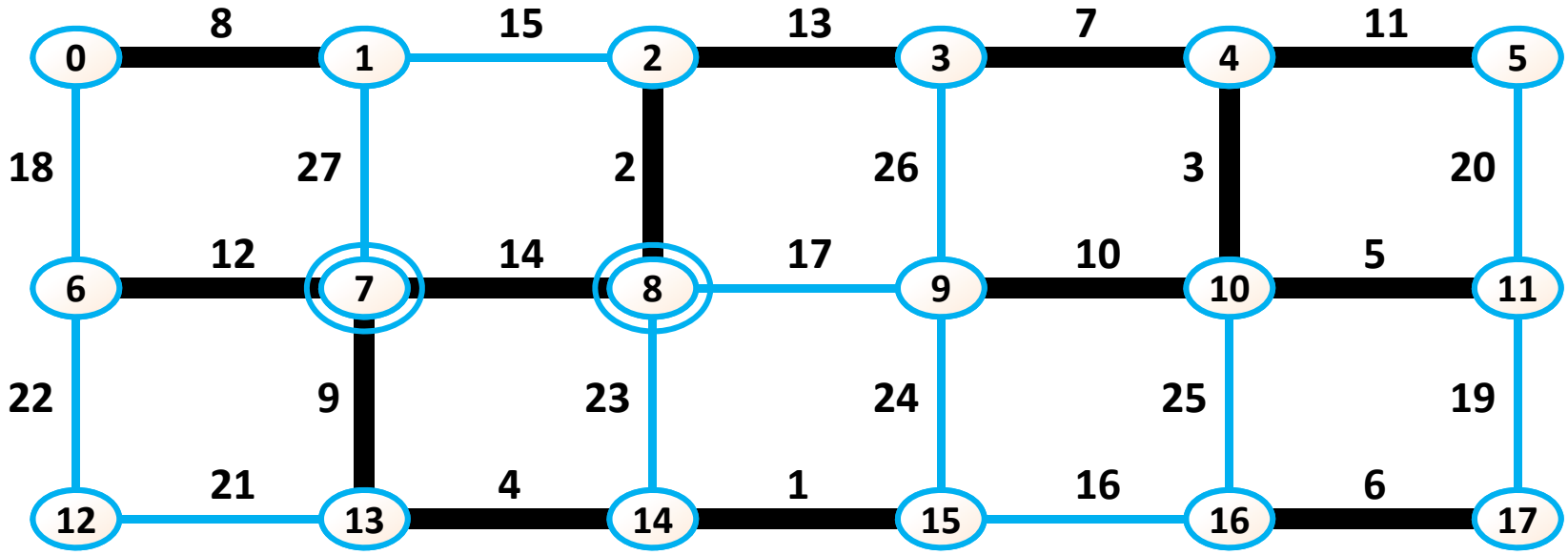
Kruskal algorithm and Union-Find scheme example

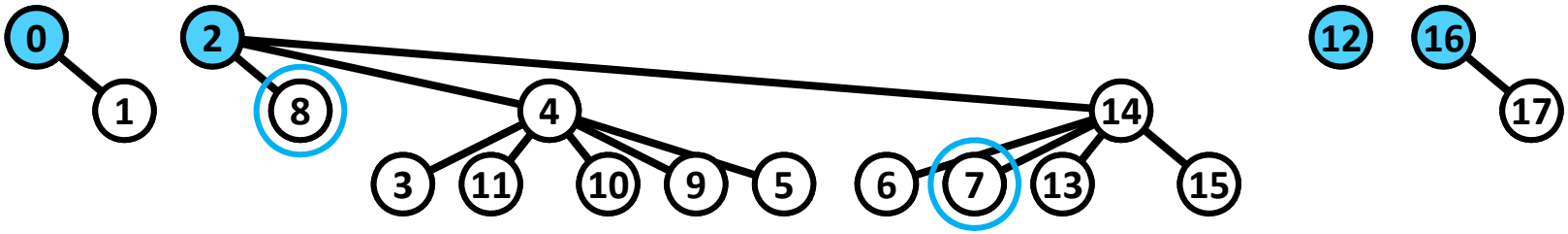




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	2	4	14	14	2	4	4	4	12	14	14	14	16	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

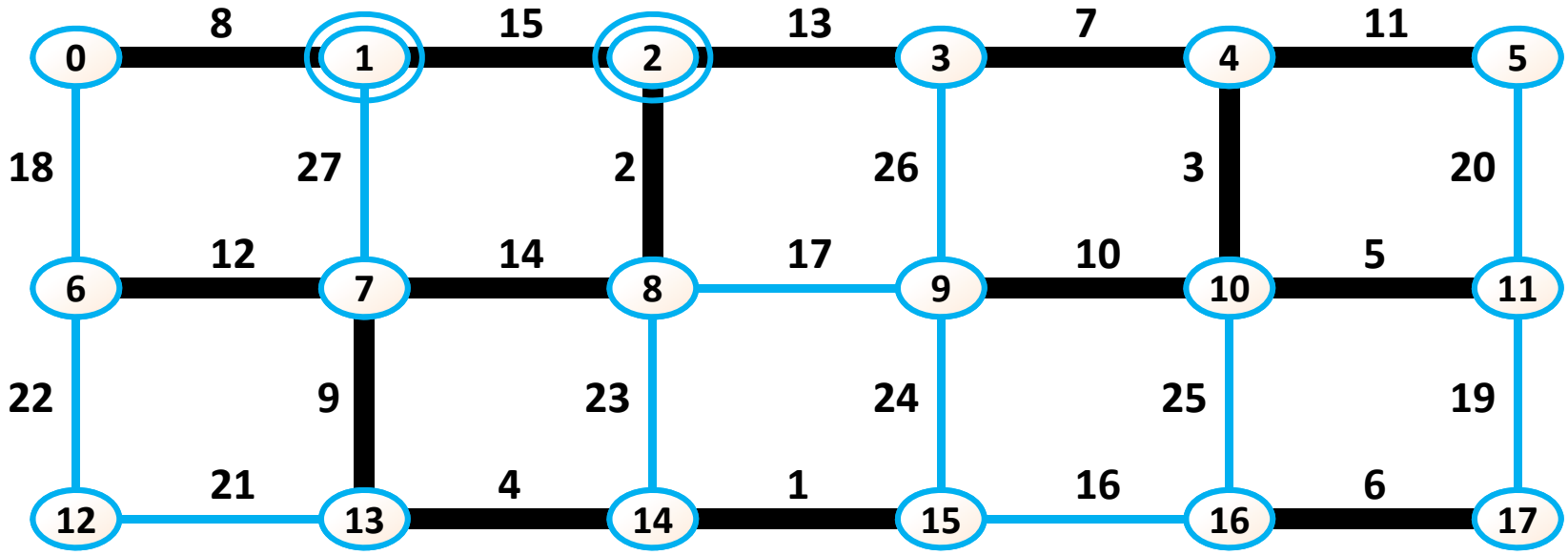
Kruskal algorithm and Union-Find scheme example

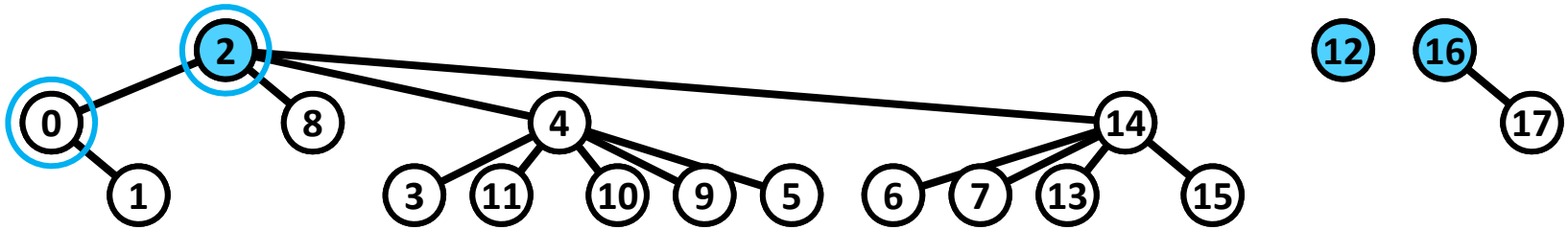




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	2	4	14	14	2	4	4	4	12	14	2	14	16	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

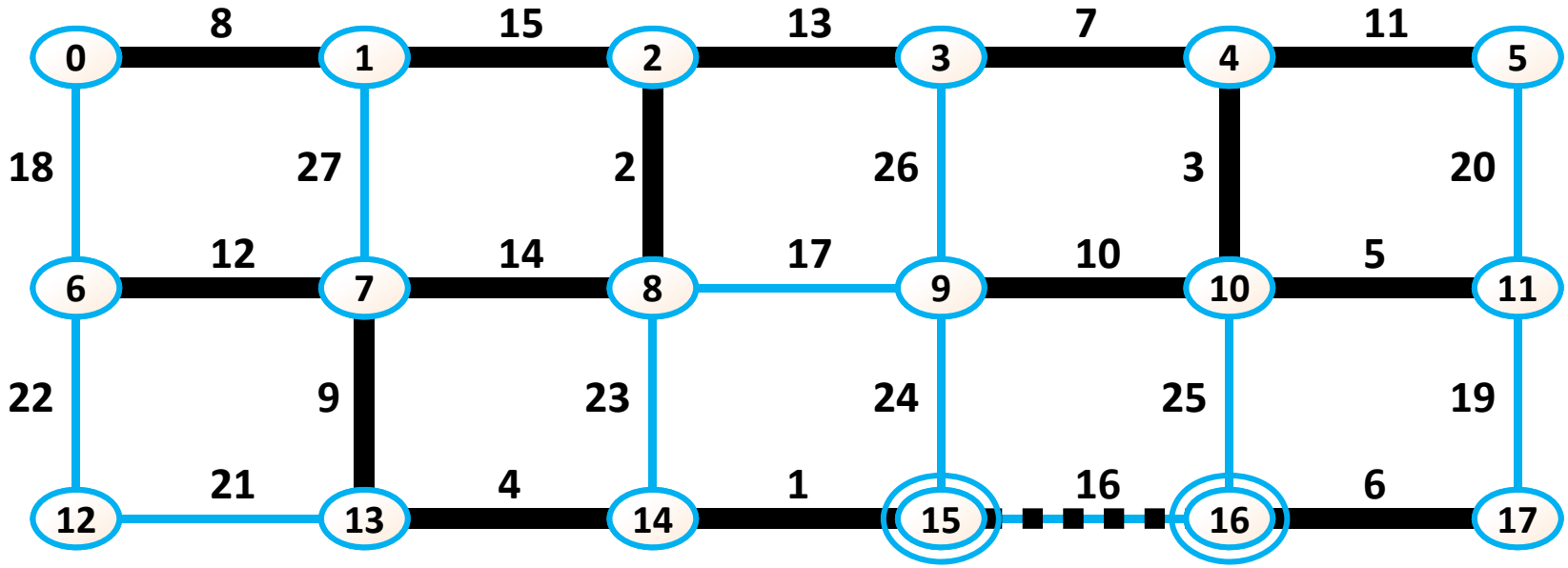
Kruskal algorithm and Union-Find scheme example

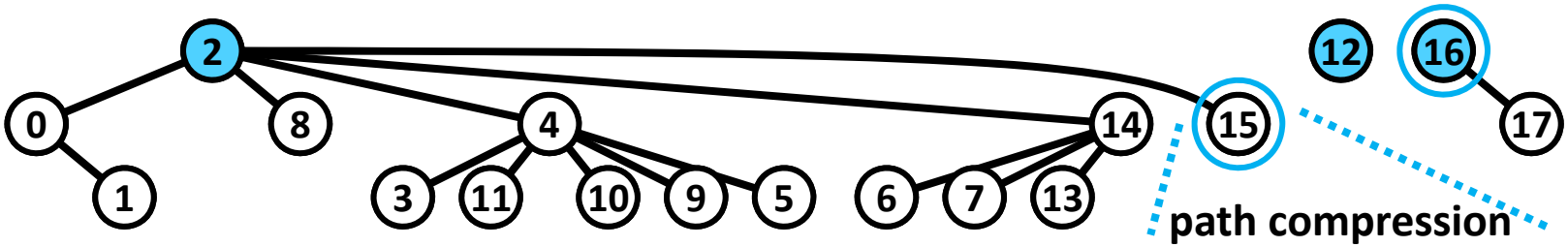




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	14	14	2	4	4	4	12	14	2	14	16	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

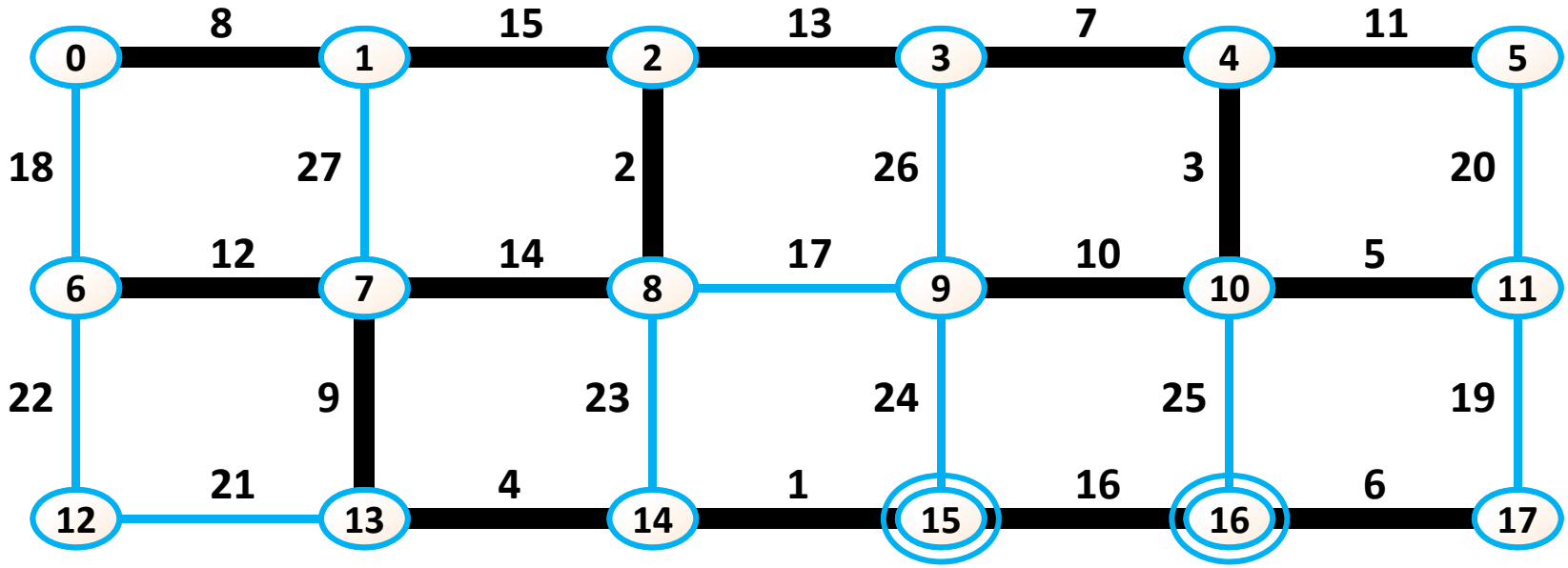
Kruskal algorithm and Union-Find scheme example

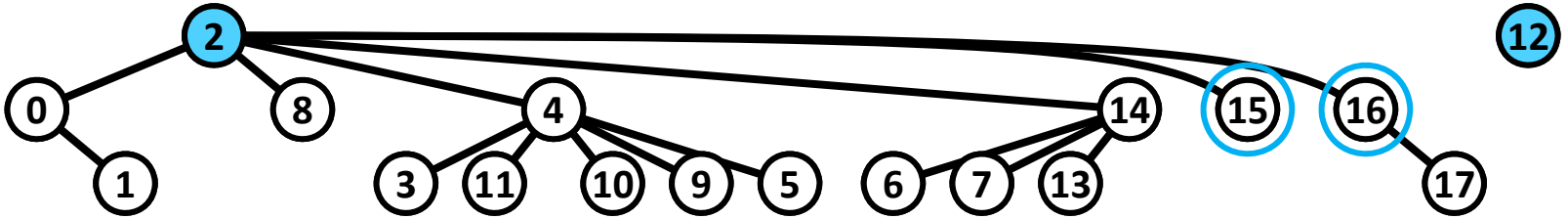




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	14	14	2	4	4	4	12	14	2	2	16	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

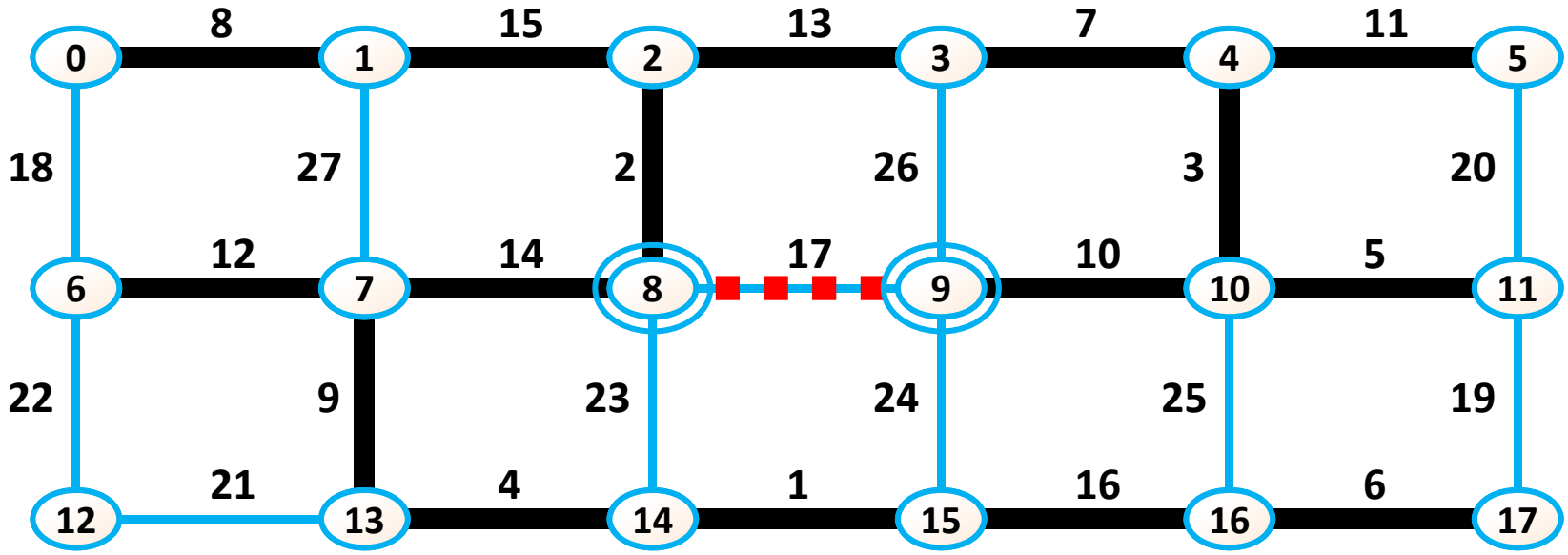
Kruskal algorithm and Union-Find scheme example

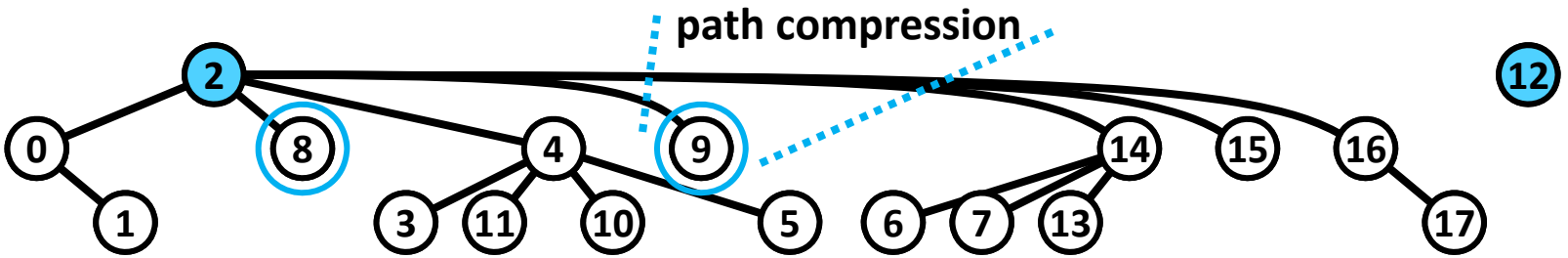




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	14	14	2	4	4	4	12	14	2	2	2	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

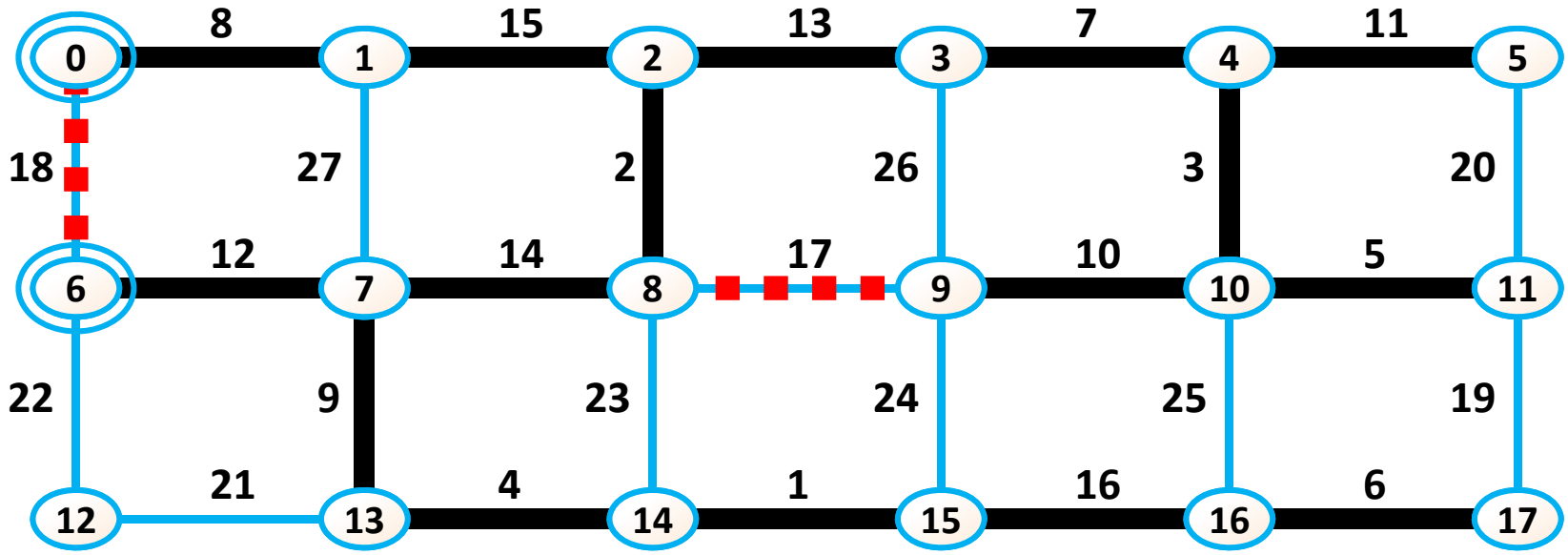
Kruskal algorithm and Union-Find scheme example

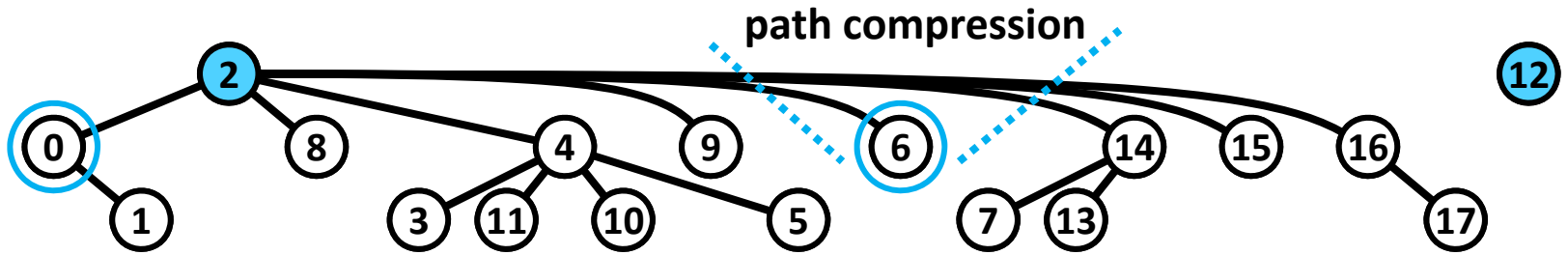




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	14	14	2	2	4	4	12	14	2	2	2	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

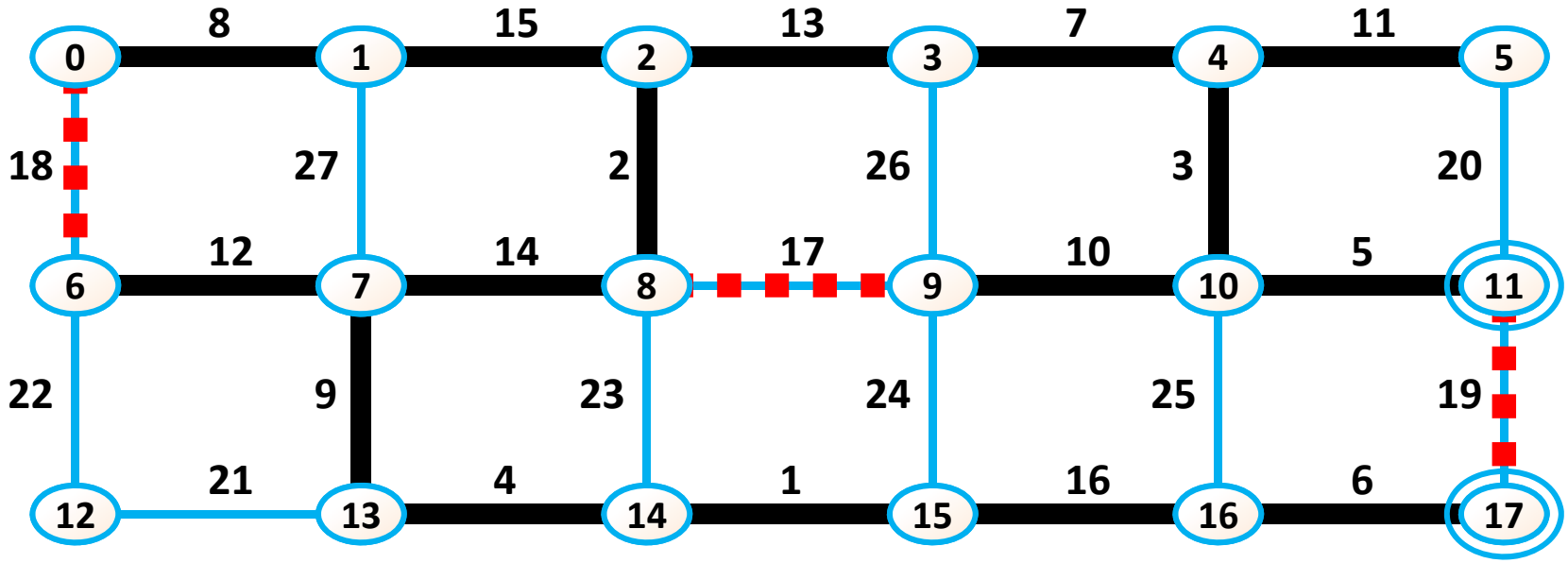
Kruskal algorithm and Union-Find scheme example

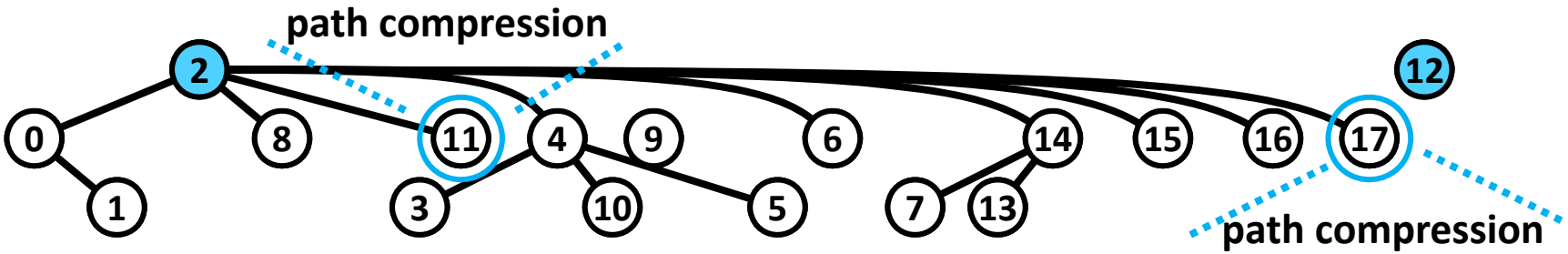




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	2	14	2	2	4	4	12	14	2	2	2	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

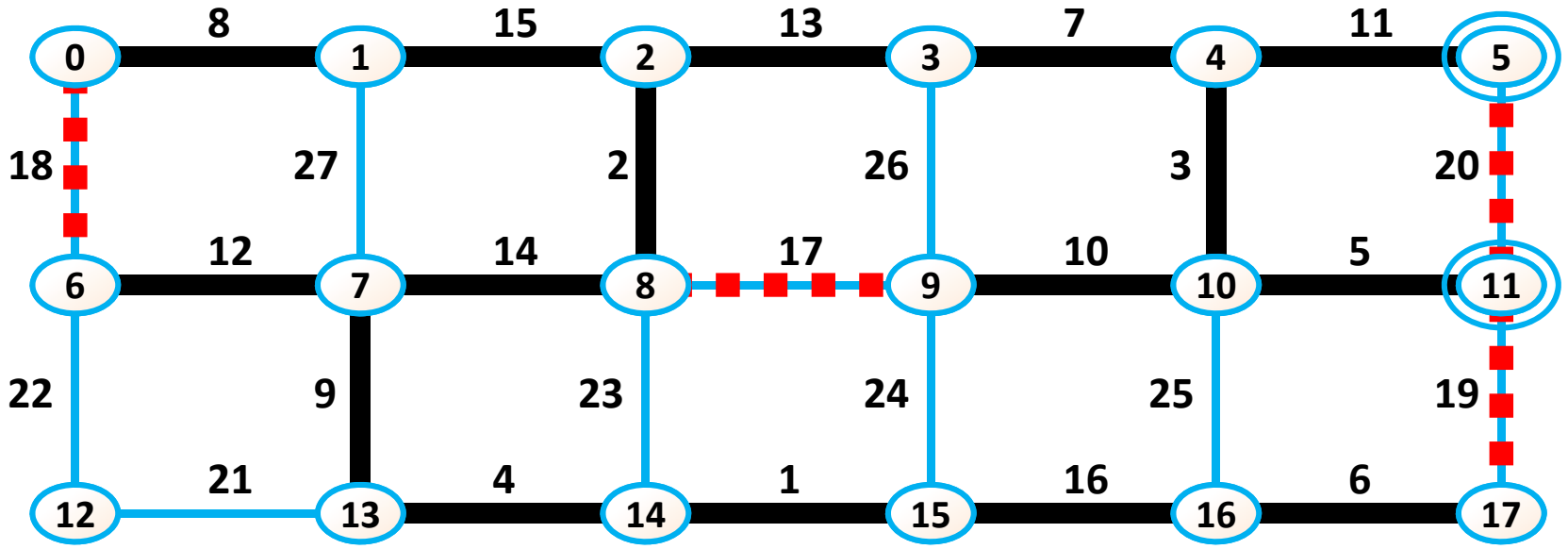
Kruskal algorithm and Union-Find scheme example

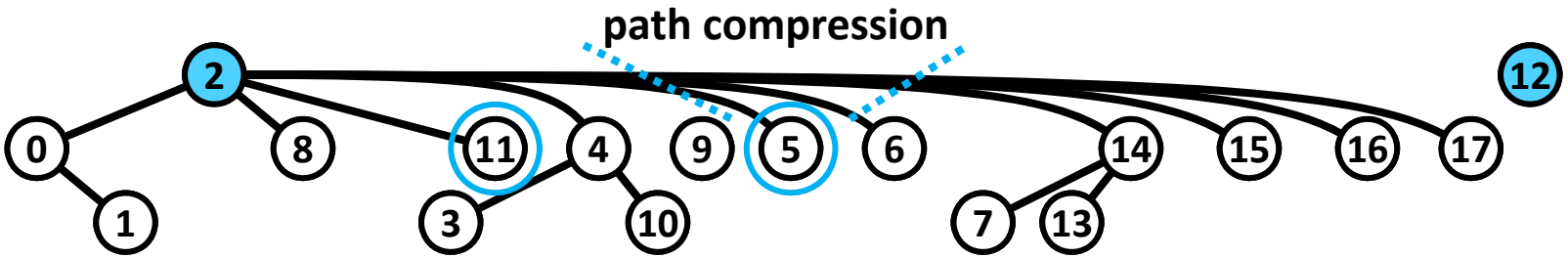




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	2	14	2	2	4	2	12	14	2	2	2	2
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

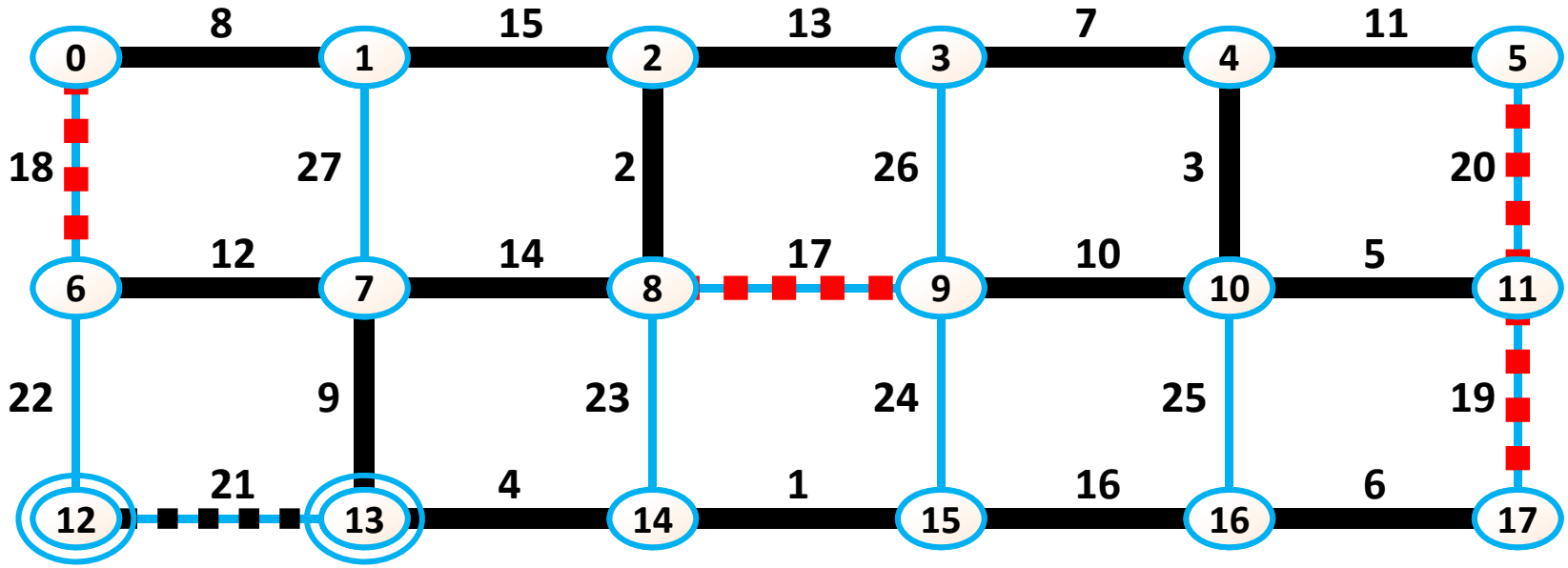
Kruskal algorithm and Union-Find scheme example

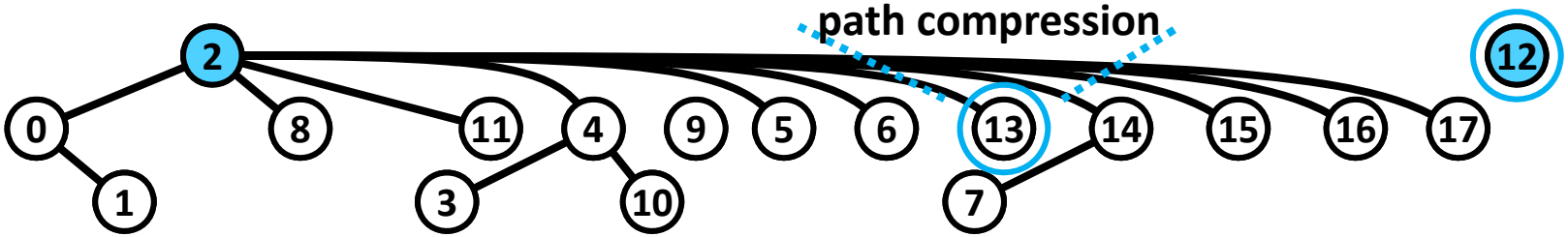




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	2	2	14	2	2	4	2	12	14	2	2	2	2
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

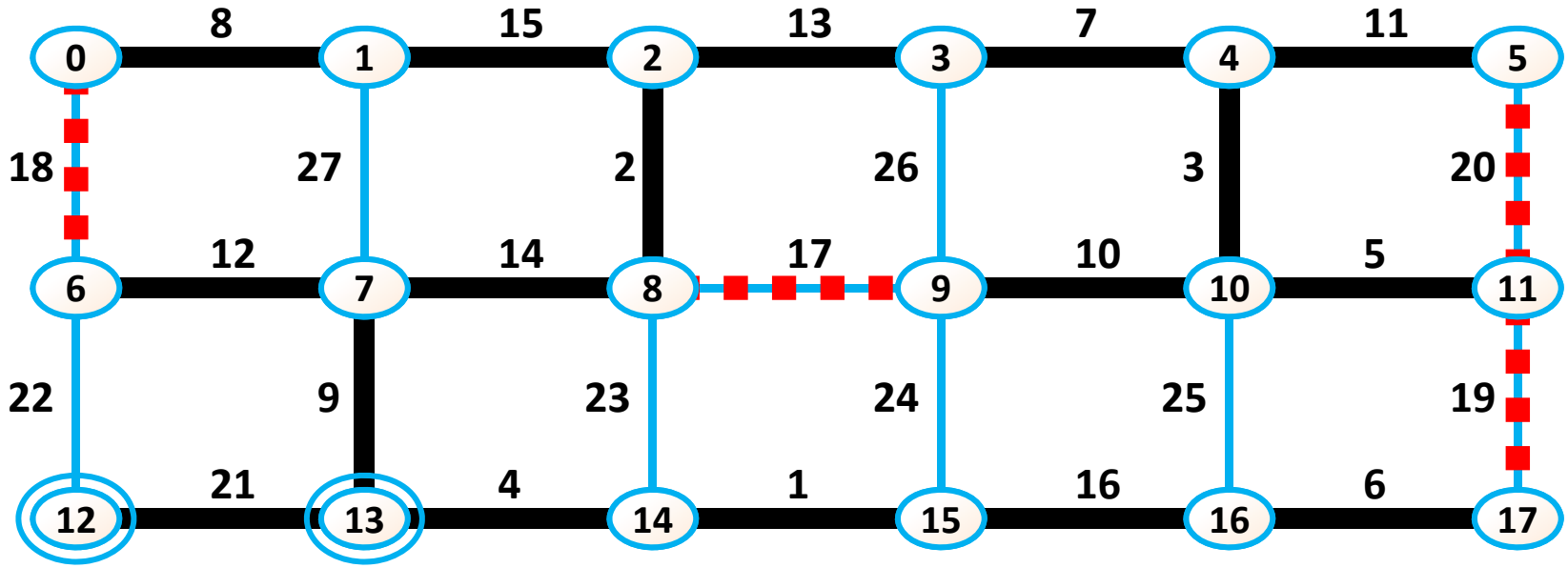
Kruskal algorithm and Union-Find scheme example

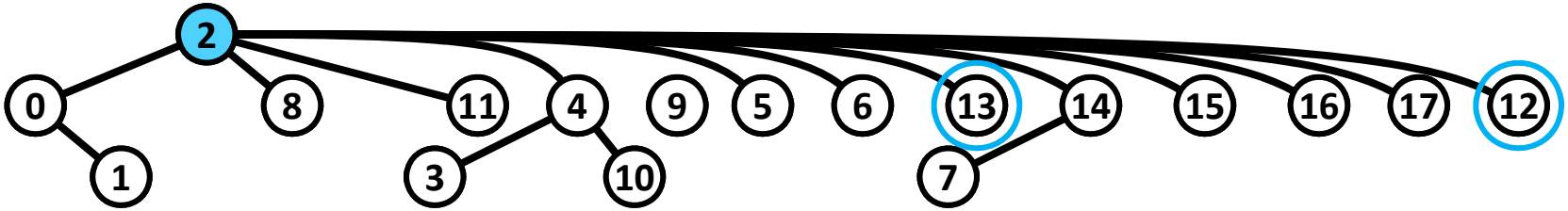




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	2	2	14	2	2	4	2	12	2	2	2	2	2
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

Kruskal algorithm and Union-Find scheme example





node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	2	2	14	2	2	4	2	2	2	2	2	2	2
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

Kruskal algorithm running time improvement

When both union by rank and path compression are used, the running time spent on Union-Find operations in a graph with N nodes and M edges is $O(M \cdot \alpha(N))$, where $\alpha(N)$ is the inverse function of $f(x) = A(x, x)$, where $A(x, y)$, is the Ackermann function, known to grow quite fast. In fact, for *any* graph representable in *any* conceivable machine $\alpha(N) < 4$.

Thanks to the inverse Ackermann function, all Union-Find operations run in amortized constant time in all practical situations.
It means that the speed bottleneck is the initial edge sorting.
Sorting can be done in linear time when:

- edge values are strings (does not happen too often), apply **Radix sort**,
- edge values are integers in some (relatively) moderate range (e.g 0..10 000, etc.), apply **Counting sort**,
- edge values are floats (more or less) uniformly distributed over some interval, apply **Bucket sort**.

Conclusion
In many practical situations, a careful implementation of Kruskal algorithm runs in $\Theta(M)$ time.

Ackermann function

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0, \\ A(m-1,1) & \text{if } m > 0, n = 0, \\ A(m-1, A(m,n-1)) & \text{if } m > 0, n > 0. \end{cases}$$

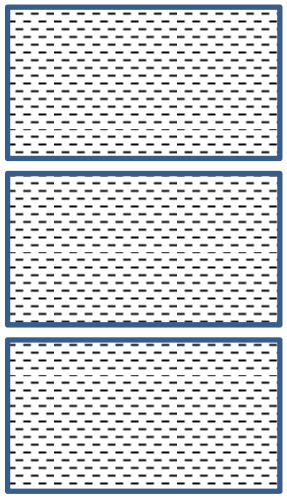
$m \backslash n$	0	1	2	3	4	5	n
0	1	2	3	4	5	6	$n+1$
1	2	3	4	5	6	7	$n+2$
2	3	5	7	9	11	13	$2n-3$
3	5	$13 = 2^4 - 3$	$= 2^5 - 3 = 29$	$= 2^6 - 3 = 61$	$= 2^7 - 3 = 125$	$= 2^8 - 3 = 253$	$2^{n+3} - 3$
4	$13 = 2^{2^2} - 3$	$65533 = 2^{2^{2^2}} - 3$	$(\#) = 2^{2^{2^{2^2}}} - 3$	$(\#\#) = 2^{2^{2^{2^{2^2}}}} - 3$	$(\#\#\#) = 2^{2^{2^{2^{2^{2^2}}}}} - 3$	$(\#\#\#\#) = 2^{2^{2^{2^{2^{2^{2^2}}}}}} - 3$	$2^{2^{2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}}} - 3$ } $n+3$
5	$65533 = 2^{2^{2^2}} - 3$	$2^{2^{2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}}} - 3$ } 65536					*too big to fit here …*
6	$A(6,0) = A(5,1)$						*too big to fit here …*
7							

$2^{2^{2^{2^{2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}}}}} - 3$ } $2^{2^{2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}}}$ } 65536

1. 0-th value in each line = 1st value in the previous line.
2. Value X in the j-th column (j > 0) on the current line is equal to the value on the previous line which column index is equal to the value written to left of the value X on the current line (= in the (j–1)-th column).

$$A(4,2) = (\#) = 2^{2^{2^{2}}} - 3 = 2^{2^{2^{4}}} - 3 = 2^{2^{16}} - 3 = 2^{65536} - 3 =$$

= (next 3 slides with 19729 decimal digits) =



$$A(4,3) = (\#\#) = 2^{(\#)} - 3 = ?$$

$$A(4,4) = (\#\#\#) = 2^{2^{(\#)}} - 3 = ??$$

Informal discussions concerning big integers:

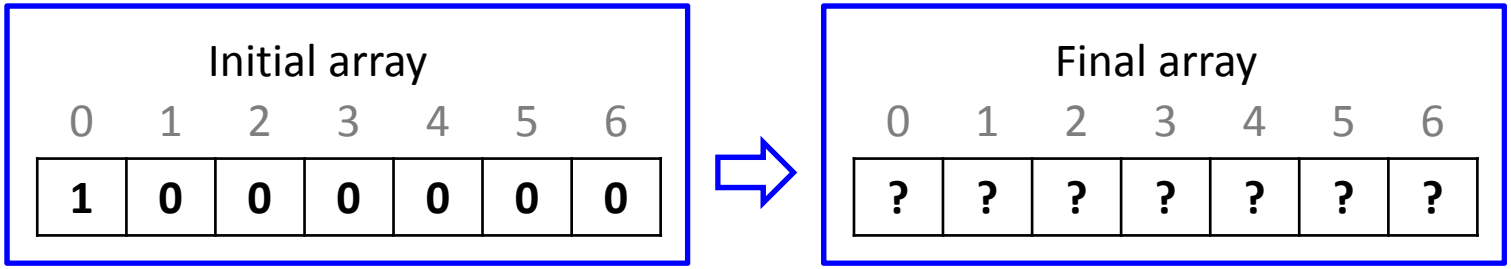
http://waitbutwhy.com/2014/11/1000000-grahams-number.html
http://www.scottaaronson.com/writings/bignumbers.html

```
200352993040684646497907235156025575044782547556975141926501697371089405955631145308950613088093334810103823434290726318182294938211881266886950636476154702916504187191635158796634721944293092798204830910485599057015931895963952486337236720300291696959215610876494888925409080591145703767520850020667156370236612635974714480711177481588091413574272096719015183628256061809145885269982614142503012339110827360384376787644904320596037912449090570756031403507616256247603186379312648470374378295497561377098160461441330869211810248595915238019533103029216280016056867010565164675056803874152946384224484529253736144253361437372908830379460127472495841486491593064725201515569392262818069165079638106413227530726714399815850881129262890113423778270556742108007006528396332215507783121428855167555407334510721311242739956298271976915005488390522380435704584819795639315785351001899200002414196370681355984046403947219401606951769015611972698233789001764151719005113346630689814021938348143542638730653955296969138802415816185956110064036211979610185953480278716720012260464249238511139340046435162386756707874525946467090388654774348321789701276445552940909202195958575162297333357615955239488529757995402847194352991354376370598692891375715374000198639433246489005254310662966916524341917469138963247656028941519977547770313806478134230959619096065459130089018888758808473362595606544488850144733570605885708016210849971452956834406197969056546981363116205379369791403236328496233046421066136200220175787851857409162050489711781820400187282939943446186224328009837323764931814789848119452713007440220765680910376203999203492023906626264491909167985461515778839060397720759279378852241294301017458086862263369284725851403039615558564330385450688652213114813638408384778263790459607186876728509763471271988890680478243230394718650525660978150729861141430305816927924971409161059417185352275887504477592218301158780701975535722241400019548102005661773589781499532325208589753463547007786690406429016763808161740550405117670093673202804549339027992491867306539931640720492238474815280619166900933805732120816350707634351669869625020969023162859350071874190579161241536897514808261904847946571736601005892476655445840838334790544144817684255327207315586349347605137419779525190365032198020108764738368682531025183377533908861426184800374008082238104076468878471647552945326947661700424461063311238021134588694532200116564076327023074292426051582811070387018345324567635625951430032037432740780879056283663406965030844225855967039271869461158513793386475699748568670079823960604393478850861649260304945061743412365828352144806726676841807083754862211408236579802961200027441324438432402331257403545019352428776430880232850855886089962774458164680857875115807014743763867976955049991643998284357290415378143438847303484261903388841494031366139854257635577105335580206622185577060082551288893332226436281984838613239570676191409638533832374343758830859233722284644287996245605476932428998432652677378373173288063210753211238680604674708428051166488709084770291208161104912555598322366244868506641402684641209694982590565519216188104341226838996283071654868525536914850299539675503954938371853405900096187489473992880432496373165753803673586710175783994818471798498246948060532081996066183434012476096639519778021441199752546704080608499344178256285092726523709898651539462193004607364507926212975917698293892367015170992091531567814439791248475706237804600009918293321306880570046591458387208088016887445835557926258465124763087148566313528934166117490617526671492672176128330845273936469244582892571388878390563004824837998396920292222154861459023734782226825216399574408017271441461795592261750838890200741699262383002822862492841826712434057514241885699942723316069987129868827718206172144531425749440150661394631691976291815065797455262361912248480638900336690743659892263495641146655030629659601997206362026035219177674066877746354937531889958786628212546979710206574723272137291814466665942187200347450894283091153518927111428710837615922238027660532782335166155514936937577846667014571797190122711781278045024002638475878833939681796295069079881712169068692953824852983002347606845411417813911064856023654975422749723100761513187002405391051091381784372179142252858743209852495787803468370333781842144401713868812424998441861812927119853331538256732187042153063119774853521467095533462633661086466733229240987984925669110951614361860154890974024191350962304361219612816595051866602203071561368473236466086890501426391390651506390819937885231836505989729912540447944342516677429965981184923315155527288327402835268844240875281128328998062591267369954624734154333350014723143061275039030739713525206933817384332295070104906186753943313078479801565513038475815568523621801041965025559618193498631591323303609646190599023611268119602344184336333459492763194610171665291382371718239429921627253846177606569454229787707138319881703696458868981186321097690035573588462446483570629145305275710127887202796536447972402540544813274839179412882642383517194919720979714593688753719872913083173803391101612854741537737715951728084111627597186384924222802373441925469991983672192131287035585307966942713416391033882754318613643490100943197409047331014476299861725424423355612237435715825933382804986243892498222780715951762757847109475119033482241412025182688713728193104253478196128440176479531505057110722974314569915223451643121848657575786528197564843508958384722923534559464521215831657751471298708225909292655638836651120681943836904116252668710044560243704200663709001941185557160472044643696932850060046928140507119069261393993902735534545567470314903886022024639948260501762431969305640666366626090207048887438898907498152865444381862917382901051820869936826618683039152732645812867828066013375000965933646251460917231803129303478774212346791184547913111098977946482169225056293999567934838016991574397005375421344858745868560472867510654233418938390991105864655951136460610551568385412174598018071331636125730796111683436376766730735458349478978831633012924080083635682593915711313097803051644171668251834657367593419808495894794098329250008638977856349469321247342610306271374507728615692259662857385790553324064184901845132828463270926975383086730840914224765947443997334813081098639941737978965701068702673416196719659159958853783482298827012560584236558953969030647496558414798131099715754204325639577607048510088157829140825077773855979012912940730946278594450585941227319481275322515232480150346651904822896140664689030510251091623777044848623022948896671138055560795662073244 9
```


373374027836767300203011615227008921843515652121379215748206859356920790214502277133099987729459596952817044582181956080965811702798062669891205061560742325686842271306295009864421853470810407128917646906550836129916694778023822502789667843489199409657361704586786242554006942516693979292624714524945408858422726153755260071904336329196375775021760051958006938476357895868784895368721228985578068265181927036320994801558744555751753127364714212955364940843855866152080121150790750685533444892586932838596530132720469706945715469593536585717888948623332924652027358531885333709484554033365653569881725825289180566354883637437933484118455801683318276768346462919956055134700391478768086403226296166415606675081537106467231084619642475374905537448053182260027102164009805844975260230356400380834720531499411729657367850664214008426964971032419191821212132069397691439233683747092282677387081322366800869247034915868409911530983154120635661231875043054675369832308279664574176208065931772656858416818379661061449634325441117069417002226578173583512598210807691019610522292638797450490192543119006205619065774524161919131875339840493439768233102984658933183730158095925228292068208622303325852801192664963144413164427730032377922747123306964171499455322610354751456312906688543454268697884477429817774937101176146516241836166802548152963353084908499430067636548061029400946937506098455885580439704859144495844507997849704558355068540874516331646411808312307970438984919050658758642581073842242059119194167418249045270028826398305795005734171148703118714283418449915345670291528010448514517605530697144176136858238410278765932466268997841831962031226242117739147720800488357833356920453393595325456489702855858973550575123512953654050284208102278524877660357424636667314868027948605244578267362623085297826505711462484659591421027812278894144816399497388188462276824485162205181707672216986326570165431691974265123004175732990447353767253684579275436541282655358185804684006936771860502007054724754840080553042495185449526724726134731817474218007857469346544713603697588411802940803961674694628854067917213860122541950381970453841726800639882065632879283958270851091995883944829777564715202613287108952616341770715164289948795356485455355314875497813400996485449863582484769059003311696130376612792346432312970662841130742704620203201336835038542536031363676357521260470742531120923340283748294945310472741896928727557202761527226828337674139342565265328306846999759709775000556088993268502504921288406827413988163154045649035077587160074055685724021758685439053228133770707415830756269628316955687424060527726485853050611356384851965918968649596335568216975437621430778665934730450164822432964891270709898076676625671517269062058815549666382573829274182082278960684488222983394816670984039024283514306813767253460126007269262969468672750794346190439996618979611928750519442356402644303271737341591281496056168353988188569484045342311424613559925272330064881627466723523751234311893442118885085079358163848994487544756331689213869675574302737953785262542329024881047181939037220666896702204258836895840939998453560948869946833852579675161882159410981624918741813364726965123980677561947912557957446471427868624053750576104204267149366084980238274680575982591331006919941904651906531171908926077949119217946407355129633864523035673345588033313197080365457184791550432654899559705862888286866606618021882248602144999973122164138170653480175510438406624412822803616648904257377640956326482825258407669045608439490325290526337532316509087681336614242398309530806549661879381949120033919489494065132398816642080088395554942237096734840072642705701165089075196155370186264797456381187856175457113400473810762763014953309735174180655479112660938034311378532532888353335202493436597912934128485497094682632907583019307266533778255931433111096384805394085928398890779621047984791968687653998747709591278872747587443980677982496827827220092644994455938041460877064194181044075826980568803894965461658798390466058764534181028990719429302177451997610449504319684150345551404482092893337865736305283061999007774872692299860827905317169187657886090894181705799340489021844155979109267686279659758395248392673488363474565168701616624064242424122896111801061568234253939218005248345472377921991122859591419187749179382334001007812832650671028178139602912091472010094787875255126337288422235386949006792766451163475810119387531965724212147603828477477457170457861041738574791130190858387789015233434301300528279703858035981518292960030568261209195094373725454171056383887047528950563961029843641360935641632589408137981511693338619797339821670761004607980096016024823096943043806956620123213650140549586250615282588033022908385812478469315720323233601899469437647726721879376826431828382603564520699468630216048874528424363593558622333506235945002890558581611275341783750455936126130852640828051213873177490200249552738734585956405160830583053770732533971552620444705429573538361113677523169972740292941674204423248113875075631319078272188864053374694213842169928862940479635305150560788126366206497231257579019598873041195626227343728900516561111094111745277965482790471250581999077498063821559376885546498822938985408291325129076478386322494781016753491693489288104203015610283386143827378160946341335383578340765314321417150655877547820252454780657301342277470616744241968952613164274104695474621483756288299771804186785084546965619150908695874251184435837306590951460980451247409411373899927822492983367796011015387096129749705566301637307202750734759922943792393824427421186158236161317886392553095117188421298508307238259729144142251579403883011359083331651858234967221259621812507058113759495525022747274674369887131926670769299199084467161228738858457586422726573330753735572823951616964175198675012681745429323738294143824814377139861906716657572945807804820559511881687188075212971832636442155336787751274766940790117057509819575084563565217389544179875074523854455200133572033332379895074393905312918212255259833790909463630202185353848854825062897715616963860712382771725621313460549401770413581731931763370136332252819127547191443450920711848838366818174263342949611870091503049165339464763717766439120798347494627397822171502090670190302469762151278521956142070806461631373236517853976292092025500288962012970141379640038055734949269073535145961208674796547733692958773628635660143767964038430796864138563447801328261284589184898528048048844180821639423974014362903481665458114454366460032490618763039502356402044530748210241366895196644221339200757479128683805175150634662569391937740237

```
83512075666260829890491877287833852178522792045771846965855278790447562192663992008409302075673925363735628390829817577902153202106409617373
28359849406665214119818381088451545977289516457213189779790749194101314836854463961690460703010759681893374121757598816512700076126278916951
04063158576375347874200702220510708912576123616580268068158584998526314658780866168007332646768302063916972030648944056281954061906852420030
53463156621891327309069687353181641094514288036605995220248248886711554429104721929134248346438705368508648749099178812670565665387191049721
82004237149274016446094345984539253670613221061653308566202118896823400575267548610147699368873820958455221157192347968688816085363161586288
01503959494185294892270744108282071693033878180849362040182552222710109856534448172074707560192459155994310729495781978785905789400525401228
67517142511184356437184053563024181225473266093302710397968091064939272722683035410467632591355279683837705019855234621222858410557119921731
71796980433931770775075562705604783177984444763756025463703336924711422081551997369137197516324130274871219986340454824852457011855334267526
47159783107312456634298052214554941562527240289153333543493412178620370072603152798707718724912344944771479095207347613854254853115527733010
30342476835865496093722324007154518129732692081058424090557725645803681462234493189708138897143299831347617799679712453782310703739151473878
69211918756670031932128189680332269659445928621060743882741691946516226763254066507088107103039417886056489376981673415902592519461182364294
56526693722031555047002135988462927580125277154220166299548631303249123110296279237238997664168034971412265279319076363261368141455163766565
59839788489381733082668779901962886932296597379951931621187215455287394170243669885593888793316744533363119541518404088283815193421234122820
03095031334105070476015998798547252919066522247931971544033179483683737322082188577334162385644138070054191353024594391350255453188645479625
22602517629283743304651023610575835145507394433396102162296754614157811271970017386114942795014112532806212547758105129720884652631580948066
33687670147310733540717710876615935856814098212967730759197382973441445256688770855324570888958320993823432102718224114763732791357568615421
25284965790333509315277692550584564401055219264450531207375628774499816364633283581614033017581396735942732769044892036188038675495575180689
00585329272014939235005258451467069826285482578832673987352204572282392902071448222198855871028969919358730742778151597576207640239512438602
02032596596250212578349957710085626386118233813318509014686577064010676278617583772772895892746039403930337271873850536912957126715066896688
49388088514294360996201296675907922508227531381284985152690293170026313632894209579757795932763553116206675348865131732387243874806351331451
26448899675898288129254800764251865864902411111273013571971813816025831785069322440079986566353715440884548663931817083957357807990597308390
94881804060935959190907473960904410150516321749681412100765719177483767355751000733616922386537429079457803200042337452807566153042929014495
78062963413838355178359976470885134900485697369796523869584599459559209070905895689145114141268450546211794502661175016692826025095077077821
19504326173832235624376017767993627960993689751913949650333585071554184364568526166742436889203710374953284259271316105378349807407391586338
17967658425258036737206469351248652238481341663808061505704829059890696451936440018597120425723007316410009916987524260377362177763430621616
74488493081092990100951797454156425120482208671458684925513244426677127863728211331536224301091824391243380214046242223349153559516890816 28
84879899882736304453724321742802157557779670216663170479697281724833928410156422745072717792693999297403080727703950135815451424940490265361
05825409373114653104943382484379718606937214444600826798002471229489405761853892203425608302697052876621377373594394224114707074072902725461
30735854174569141944648762435768239706570318416846754073346634629367398362000404140071405427763248013274220268539369886978760700959004868465
06267713630709798210065572851013066010107806337433447730734786538817426812307437660666433127753564665786037151929227684404582732832438082128
41218776132042460464900801054731426749260826922155637405486241717031027919996942645620955619816454547662045022411449404749349832206807191352
76798674781345820385957041346617793722853494003163159954409368408957253343870298671782977037333280680176463950209002394193149911500910527682
11195109990631661503115855828355826071794100525285836113699613034427901738117874120612881820620232638498615156564512300477929675636183457681
05043341769543067538041113928553792529241347339481050532025708728186307291158911335942014761872664291564036371927602306283840650425441742335
46454998705531872688792642410214736369862546374715974435494344389973005174252511087735788639094681209667342815258591992485764048805507132981
42993599114632399191139599267525763590074465728101918058418073422277347213977232182317717169164001088261125490933611867805757223910181861685
49108500885272274374212086524852372456248697662245384819298671129452945515497030585919307198497105414181636968976131126744027009648667545934
56705993699546450055892162804797636568613331656390739570327203438917541526750091501119885687270884819553167693168127289214303137681801644547
73675183534978579242764633541624336011259602521095016122641103460834656482355979342740568688492244587454937767521203247038030354911575448312
95275891939893680876327685438769557694881422844311998595700727521393176837831770339130423060958999137314684569010422095161967070506420256733
87344611565527617599272715187766001023894476053978951694570880272873622512107622409181006670088347473760515628553394356584375627124124445765
16630640859395079475509204639322452025354636344447917556617259621871992791865754908578529500128402290350615149373101070094461510116137124237
61426722541732055959202782129325725947146417224977321316381845326555279604270541871496236585252458648933254145062642337885651464670604298564
78196846159366328895429978072254226479040061601975197500746054515006029180663827149701611098795133663377137843441619405312144529185518013657
55586676150193730296919320761200092550650815832755084993407687972523699870235679310268041367457189566414318526790547171699629903630155456450
90044802789055701968328313630718997699153166679208958768572290600915472919636381673596673959975710326015571920237348580521128117458610065152
59888384311451189488055212914577569914657753004138471712457796504817585639507289533753975582208777750607233944558789590571915 6733
```

A game with some relation to the Ackermann function



On a table:

Init: Array with 7 fields, leftmost field contains 1 token, other fields are empty.

In each step:
-- Remove a token from any field F
-- Choose one of the following:
- If there is a field to the right of F
  add two tokens to that field
- If there are two fields to the right of F
  swap the contents of those fields

The objective:
Maximize the number of tokens in the array.

In a computer

Init: A[0] = 1; A[k] = 0, k = 1..6

In each step:
-- choose index k
-- if (A[k] > 0)
  A[k] –= 1;
  choose one of the following:
  either A[k+1] += 2; (if k < 6)
  or swap (A[k+1], A[k+2]) (if k < 5)

The objective:
Maximize sum(A[k], k=0..6).